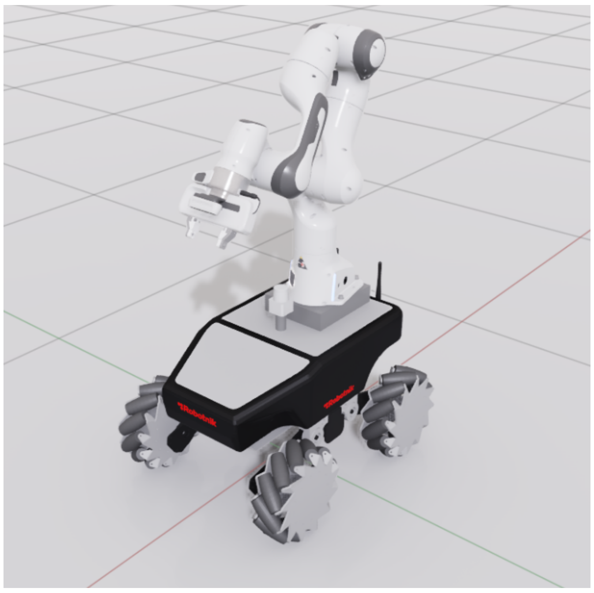
Simulation Design of a Robotic Mobile Manipulator for Material Acceleration Platforms



Master thesis

Matej Loris Debijadi

Manufacturing Technology
Aalborg University
http://www.aau.dk

Title:
Simulation Design of a Robotic Mobile Manipulator for Material Acceleration Platforms

Project Period:
Spring Semester 2023

Author:
Matej Loris Debijadi

Supervisor(s):
Simon Bøgh

Copies: 1

Page Numbers: 69

Date of Completion:
June 1, 2023

Abstract:

This thesis investigates the potential of autonomous robotic systems in Material Acceleration Platforms (MAPs) and presents the implementation of an omnidirectional mobile manipulator in a laboratory setting. The thesis focuses on the practical application of the Isaac Sim simulation tool and ROS framework for controlling and simulating robotic systems. A simulated system is developed, comprising a Summit XL mobile platform and a Franka Emika Panda robot arm. Motion planning is facilitated by ROS-based frameworks, incorporating the Navigation Stack 2 for autonomous navigation and the Moveit2 platform for motion planning and trajectory execution. Through comprehensive testing, valuable insights are gained into the workflow of the system, offering opportunities for future improvements. This work serves as a basis for further research in mobile manipulators and MAPs, contributing to The Pioneer Center for Accelerating P2X Materials Discovery (CAPeX)'s overall goal of revolutionizing the discovery of new materials.

Contents

Preface **iv**

1 Introduction **1**
1.1 Material Acceleration Platforms (MAPs) 1
1.2 Autonomous Mobile Manipulators 3
1.3 Motivation 3
1.4 Challenges 4
1.5 Initial Problem Formulation 5

2 Problem Analysis **6**
2.1 Mobile Manipulator System 6
2.2 Related works 18
2.3 Summary of Analysis 24

3 Problem Formulation **25**
3.1 Final Problem Formulation 25
3.2 Task Definition 25
3.3 Project Objectives 26

4 Solution Architecture **28**
4.1 Robot Specifications 28
4.2 Robot Operating System (ROS) 30
4.3 Navigation Stack 33
4.4 Motion Planning for Robot Arm 35
4.5 Simulation Tool 35

5 Implementation **37**
5.1 Robot Description 37
5.2 Holonomic Controller 39
5.3 Occupancy Map 42
5.4 Localization and Navigation 43
5.5 Arm Control 46

5.6 System Overview 48

6 Testing 51

6.1 Navigation System 51

6.2 Robot Arm System 55

7 Discussion 60

7.1 System Design 60

7.2 Evaluation of Project Objectives 61

7.3 Future Works 62

8 Conclusion 64

Bibliography 65

A GitHub Repository 68

B Video Documentation 69

Preface

This Master's Thesis is written by Matej Loris Debijadi as his final project of MSc programme in Mechanical Engineering, with a specialization in Manufacturing Technologies, at Aalborg University. The thesis was completed during the academic year 2022/23.

I'm very thankful to Simon Bøgh for his guidance, which had a significant impact on the development of this project. Also, I would like to thank my family for their support throughout my educational journey.

Chapter 1

Introduction

Material acceleration platforms (MAPs) are an emerging paradigm for accelerating materials discovery in an effort to develop technological solutions that can help mitigate or address climate change issues. These platforms enable autonomous experimentation by combining artificial intelligence (AI), robotic systems, and high-performance computing (HPC). [22] In this thesis, the current state of MAPs and their potential applications for autonomous mobile manipulators are examined. Discussing the opportunities and challenges associated with integrating MAPs with autonomous mobile manipulators.

1.1 Material Acceleration Platforms (MAPs)

MAPs are systems that use AI to design, execute, and analyze experiments for the purpose of materials discovery [7]. They can automate the entire workflow of materials research, from generating hypotheses and selecting candidates to synthesizing and characterizing materials, evaluating and optimizing their properties [14]. In addition to learning from data and feedback, MAPs can adjust their strategies accordingly [7]. MAPs aim to reduce the time and cost of materials development and enable new discoveries beyond human intuition [14].

As seen from the figure 1.1 MAPs are a multidisciplinary approach that aims to accelerate material discovery and optimize it by integrating human expertise, AI models, high-performance computing, databases, robotic platforms, and orchestrator software.

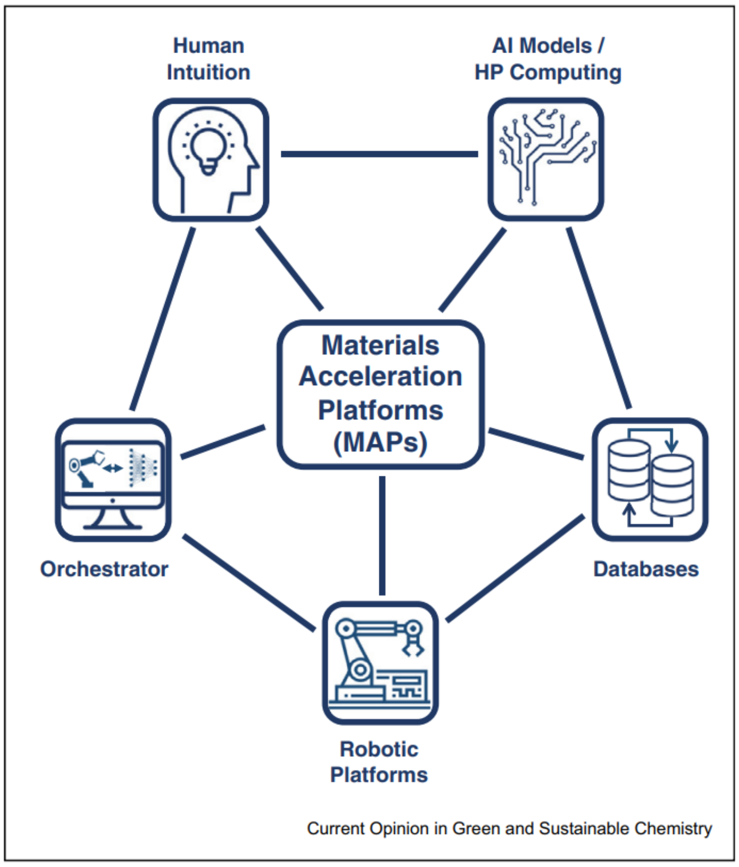

Human Intuition
AI Models / HP Computing
Materials Acceleration Platforms (MAPs)
Orchestrator
Databases
Robotic Platforms
Current Opinion in Green and Sustainable Chemistry

Figure 1.1: Main areas required to enable MAPs, that are comprised of human intuition, artificial intelligence models employing machine learning for inverse design and chemical exploration, collected databases for high-quality data, robotic platforms for automated experiments, and orchestrator software to facilitate communication between modules. [7].

Robotic platforms will be the primary technological focus of this thesis.

MAPs include robotic platforms as a vital component. These systems use automation and high-throughput techniques to conduct a large number of experiments in a short amount of time, allowing the screening of a vast chemical space in search of new materials [7].

In MAPs, robotic platforms can be utilized for a variety of tasks, including experiment preparation, automated synthesis, and characterization of materials with high precision and efficiency. These systems can be programmed to operate autonomously, conducting experiments around the clock without human intervention. By automating these tasks, researchers can rapidly screen a vast number of materials and conduct experiments that would be impractical or time-consuming to conduct manually. In addition to reducing human errors and biases, they can also improve experiment safety and scalability.

Robotic platforms can also enable the development of closed-loop systems, in which experimental results are fed back into the design process to continuously refine the search for new materials. This strategy can lead to the creation of materials with highly optimized properties, such as strength, conductivity, and durability [22].

1.2 Autonomous Mobile Manipulators

This thesis focuses on the potential implementation of autonomous mobile manipulators for MAPs. In this section, a brief introduction to AMMs and potential MAP use cases will be provided.

Autonomous mobile manipulators are robots that can navigate their environment and manipulate objects without human intervention. They use cameras, sensors, artificial intelligence, and machine vision to autonomously navigate through uncontrolled environments and perform a variety of tasks. In MAPs, autonomous mobile manipulators can perform tasks such as sample preparation, conducting experiments, and collaborating with human researchers. These tasks are necessary for exploring the vast space of potential materials and discovering novel materials with desirable properties. By employing autonomous mobile manipulators in MAPs, the advancement of materials research can be achieved in terms of productivity, precision, flexibility, scalability, safety, and innovation.

1.3 Motivation

One of the primary goals of MAPs is to reduce the time and cost of materials discovery for clean energy applications such as solar cells, batteries, catalysts, and thermoelectrics [22]. According to a report by Mission Innovation, it takes an average of 10–20 years to commercialize a new material for clean energy after its initial discovery [17]. This is too slow and costly to meet the pressing global challenges of energy security and environmental sustainability. Using data-driven methods that can explore large and complex design spaces more efficiently and intelligently than conventional approaches, MAPs aim to reduce this duration by orders of magnitude [14, 7]. A further purpose of MAPs is to increase the safety and reliability of materials research by automating dangerous or tedious tasks that are prone to human error or fatigue [14]. MAPs can, for instance, handle toxic or flammable chemicals, perform high-throughput synthesis and characterization, monitor experimental conditions and outcomes in real-time, and ensure data reproducibility and traceability [14, 7]. By assigning these tasks to machines, human researchers can concentrate on more creative and strategic aspects of materials discovery.

As part of The Pioneer Center for Accelerating P2X Materials Discovery (CAPeX)[1], researchers and students from Aalborg University aim to contribute to the CAPeX's overall goal of revolutionizing the new material discovery over the next 12 years. Therefore, the aim of this thesis is to contribute to such a case by proposing an autonomous mobile manipulator system design in a simulated laboratory setting.

1.4 Challenges

MAPs have great potential for accelerating the discovery and development of new materials. However, they must overcome a variety of challenges in order to do so.

The availability and quality of data is a major challenge. To develop accurate models and predictions, MAPs rely heavily on data, but the quality and availability of data can be a serious obstacle. Inaccurate models and prediction can result from insufficient, inconsistent, or biased data. [24]

Another difficulty in MAPs is integrating computational and experimental methods. While MAPs aim to effectively predict material properties, they must be combined with experimental methods in order to validate predictions and optimize materials. Close collaboration between computational and experimental teams is required, which can be difficult in some organizations. [13]

In addition to these challenges, integrating autonomous mobile manipulators into MAPs presents a number of new ones. One such challenge is navigation and obstacle avoidance. Such systems must be able to navigate complex environments safely while avoiding obstacles, which requires the use of advanced algorithms, sensors and path planing strategies.

Objects can be challenging to manipulate, particularly if they are fragile, have complex geometries, or are transparent. Therefore, autonomous mobile manipulators must be designed to perform diverse manipulation tasks while ensuring the safety of the materials. If the object to be manipulated is in liquid form and must be transported in a chemical flask, the mobile manipulator must be extremely stable to prevent chemical spillage.

In order to optimize their performance, mobile manipulators may be required to collaborate with other robots or humans, and must be able to effectively coordinate their actions.

Implementation of autonomous mobile manipulators present a set of challenges that must be addressed through the application of advanced technologies, algorithms, and design strategies. By overcoming these challenges, the mobile manipulators can provide a flexible and effective solution to material handling in the laboratory setting.

[1]https://www.dtu.dk/capex

1.5 Initial Problem Formulation

Given the aforementioned information, the initial problem statement that will guide the project's problem analysis is as follows:

What are the applications of autonomous mobile manipulators in Material Acceleration Platforms (MAPs), and how should such a system be designed?

Chapter 2

Problem Analysis

This chapter addresses the initial problem formulation and provides insights into the autonomous mobile manipulator system. To achieve this, the chapter conducts a detailed analysis of the system's components, dividing it into two subsystems.

Additionally, this chapter explores related works and examines the potential applications of mobile manipulators, specifically in material acceleration platforms. By providing this comprehensive review, it is hoped that the final problem statement can be formulated, and the task to be solved by the use of mobile manipulators can be defined.

2.1 Mobile Manipulator System

Robotic manipulators are fixed-base robotic systems with a limited workspace determined by the reach of their arm. To carry out tasks successfully, careful planning is necessary to ensure the robotic arm can access all the required parts. This often involves the use of conveyor belts or similar transportation systems to feed parts to the manipulator. However, advancements in mobile robots have changed this dynamic.

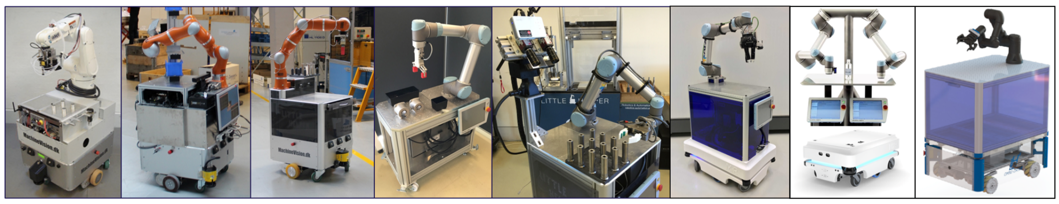


Figure 2.1: Aalborg University's Little Helpers mobile manipulators, generations one through eight (LH1 on the left and LH8 on the right) [20].

Mobile robots are controllable systems that use sensors and other technologies to perceive their environment and safely navigate through it. These robots typically incorporate physical elements such as wheels, tracks, or legs along with control algorithms. By mounting one or more robotic arms on a mobile base, a new system is created that combines the advantages of both types of robots, known as a mobile manipulator. With its ability to move around the environment, it can navigate to where the parts are located, rather than relying on a conveyor belt or other transportation system. This makes it a valuable tool in many industries, from manufacturing to logistics and beyond. [2]

In order to gain a more comprehensive understanding of the mobile manipulator system's structure, it is useful to categorize its components into two distinct subsystems: hardware and software, as illustrated in figure 2.2. Each of these components will be analyzed in further detail in the following sections.

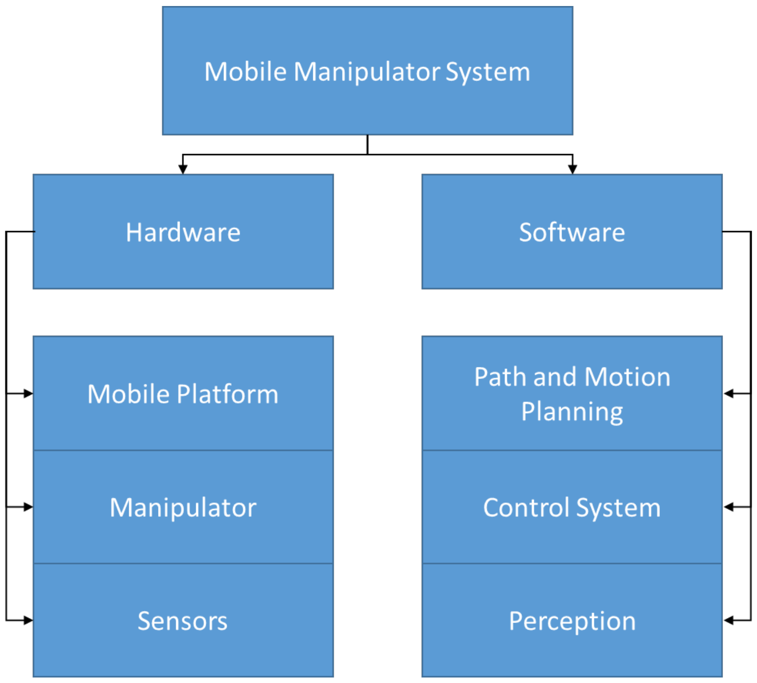


Figure 2.2: Structure and components of Mobile Manipulator System [25]

2.1.1 Hardware

The hardware of mobile manipulator system consists of three essential components: the mobile platform, the manipulator, and the sensors. The mobile platform enables the system to move, while the manipulator interacts with objects from its environment to complete specific tasks using its end-effector. The sensors are used to perceive the environment, objects, and the system itself, thus enabling safe navigation and efficient manipulation of objects. [25]

Mobile platform

Locomotion is a crucial aspect of mobile robotics that allows robots to move through their environment. Based on the locomotion type, the mobile robots can be categorized into three categories: legged, wheeled and tracked mobile robots.

Examples of different mobile platforms based on their locomotion type is shown in figure 2.3. The ANYmal C[1] legged robot, developed by AnyBotics designed for industrial inspection. The MaXXII-S[2] , created by Robodyne, is a tracked mobile robot built for diverse use cases, particularly in agriculture. Lastly, the Perseverance Mars rover [3], designed by NASA, is specialized for rock sampling.

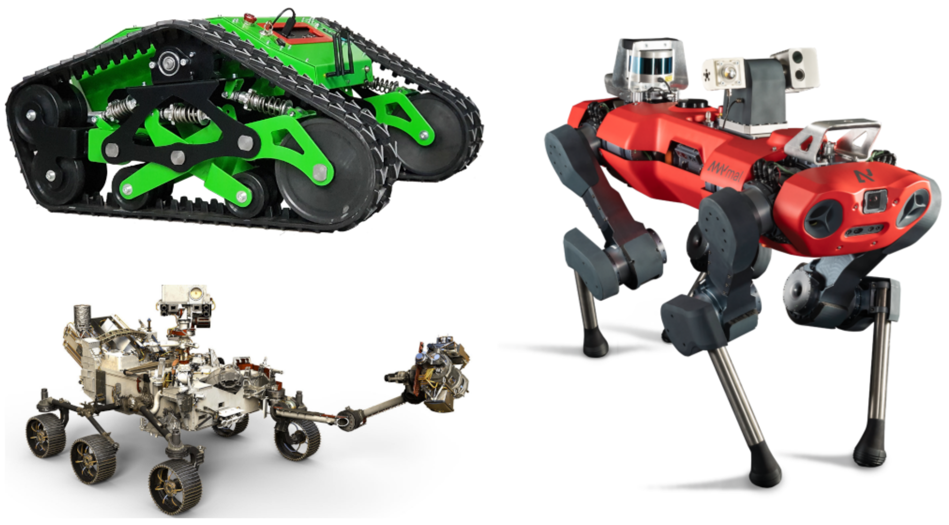


Figure 2.3: Different mobile robots with distinct locomotion types and intended applications. The ANYmal C legged robot, is shown on the right. The MaXXII-S tracked mobile robot displayed on the top left. The Perseverance Mars rover is shown on the bottom left.

Each type has certain design considerations that affect the robot's performance and capabilities. Some key design considerations for locomotion in mobile robots include stability, energy efficiency, maneuverability, and terrain adaptability. These factors can influence the choice between legged and wheeled locomotion and the specific design of the robot's legs or wheels. [19]

Legged robots offer better stability and terrain adaptability but require more energy due to the energy consumption involved in lifting and moving their legs. Additionally, legged robots may have lower maneuverability than wheeled mobile robots. On the other hand, wheeled robots are highly energy-efficient and have

[1]https://www.anybotics.com/the-next-step-in-robotic-industrial-inspection/
[2]https://robodyne-services.com/maxxii-ugv-tracked-robot-vehicle
[3]https://mars.nasa.gov/mars2020/

simpler control mechanisms than legged robots. However, they may struggle with rough or uneven terrain. Tracked mobile robots use caterpillar tracks and have larger ground contact patches, making them highly maneuverable on loose surfaces. However, they require a skidding turn to change direction, which results in power inefficiency on surfaces other than loose ones. [15, 19]

This thesis focuses on the use of mobile manipulators in MAPs, with a primary emphasis on their deployment in chemistry laboratories. Wheeled robots are often preferred over legged and tracked robots in these environments for several reasons. Firstly, the laboratory floors are flat and smooth, making them ideal for wheeled locomotion. Secondly, wheeled robots can be designed to carry heavy loads such as equipment or chemicals more easily than legged robots. Lastly, wheeled robots have simpler control mechanisms, making them easier to integrate into a laboratory automation system.

Wheeled mobile platforms Wheel design is an important factor for mobile robots that affects their stability, energy efficiency, maneuverability, and terrain adaptability. There are different types of wheels that can be used for mobile robots, each with its own advantages and disadvantages. Some of the common wheel designs are are shown in the figure 2.4.

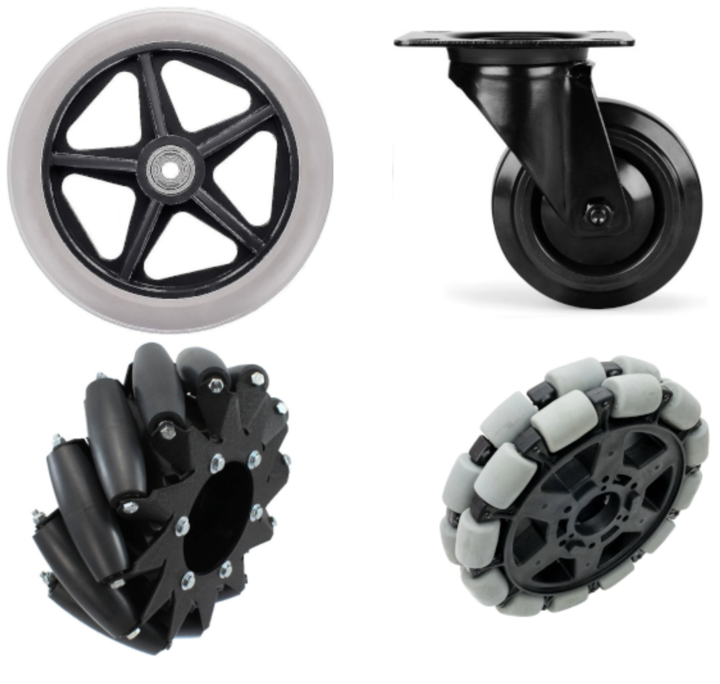


Figure 2.4: Common wheel types. Standard wheel (top left), Caster wheel (top right), Swedish or Mecanum wheel (bottom left), Omni wheel (bottom right)

Standard wheels are widely used due to their simplicity, which is limited to forward and reverse rotation. They allow the robot to rotate when different wheel rotation speeds and/or directions are applied. Steering wheels, on the other hand, have a different mechanical structure that allows them to rotate about their vertical axis. A mechanism uses a steering motor to control the direction of a wheel's movement, and a driving motor to provide forward and reverse motion. Therefore, the same physical wheel could be used as a standard wheel or converted into a steering wheel by attaching a steering mechanism. [18]

Caster wheels enable a robot to achieve near omnidirectional movement. The wheel can passively rotate 360 degrees with respect to the vertical axis as well as forward and backward, allowing the wheel to move without restriction. They do not require a steering mechanism, but their stability and traction are poor. They are suited for indoor environments but struggle on surfaces that are uneven or slippery. [19, 18]

Mecanum wheels are another name for Swedish wheels. They consist of a central hub with rollers mounted at a 45-degree angle around its circumference relative to the wheel's axis. By applying different velocity to each wheel, a robot can move in any direction. Mecanum wheels have three degrees of freedom: wheel rotation, roller rotation, and rotational slip about the vertical axis passing through the contact point of locomotion. They have a high degree of maneuverability and omnidirectional capability, but are inefficient and complex. They perform poorly on soft or rough terrains, but perform well on flat or hard surfaces. [19, 18]

Similar to Mecanum wheels, Omni wheels feature non-actuated rollers mounted on an active main wheel. And in contrast to mecanum wheels, the rollers of Omni wheels are placed at a 90-degree angle to the axis of rotation of the main wheel. [18]

The property of Holonomicity of a robot depends directly on the type of wheels used. A holonomic system has the same number of controlled DoFs as its total number of DoFs. On the other hand, a robotic system is non-holonomic if the number of its controlled DoFs is less than its total number of DoFs. The standard setup with differential drive is an example of a non-holonomic system, while a setup with Omni wheels or Mecanum wheels is an example of a holonomic system.[18]

The choice of wheel design depends on the application and environment of the mobile robot. For example, an omnidirectional robot may use Swedish wheels or Omni wheels to achieve high mobility and agility in confined spaces or dynamic scenarios. However, these wheels may not be optimal for outdoor applications where standard wheels may provide better traction and efficiency. The geometry of the wheel also affects its performance, such as its diameter, width, shape, material, etc.. Therefore, it is important to consider these factors when designing a mobile robot's locomotion system. [19]

Manipulator

Robot manipulators can be categorized into two types the traditional industrial robots and collaborative robots (cobots).

Industrial robots are large, heavy machines that operate in fixed positions on production lines. They're designed for high-speed, repetitive tasks that require precision and strength. They can lift and move heavy objects and are ideal for tasks like welding, painting, and assembly.

Cobots, on the other hand, are designed to work alongside humans in shared workspaces. They're smaller, lighter, and equipped with sensors and safety features to prevent collisions or injuries. They're more flexible than industrial robots and can be easily moved between workstations or reprogrammed for different tasks. Cobots are often used for small-batch production or assembly tasks that require human dexterity and decision-making skills. They're also easy to program, making them accessible to small businesses or manufacturers without technical expertise.

Franka Research 3 [4] cobot is shown in figure 2.5. Force-sensitive cobot designed for cutting-edge AI and robotics research. The robot has 7 degrees of freedom, a payload of 3 kg, and a maximum reach of 855 mm.

A robot manipulator's end-effector is what enables it to interact with its environment. End-effectors come in different types such as grippers, suction cups, magnets, hooks, and tools. Mobile manipulators are often designed to be modular, allowing for quick and easy end-effector changes based on the task at hand. This versatility makes mobile manipulators suitable for a wide range of applications.[4]

[4]https://www.franka.de/research/

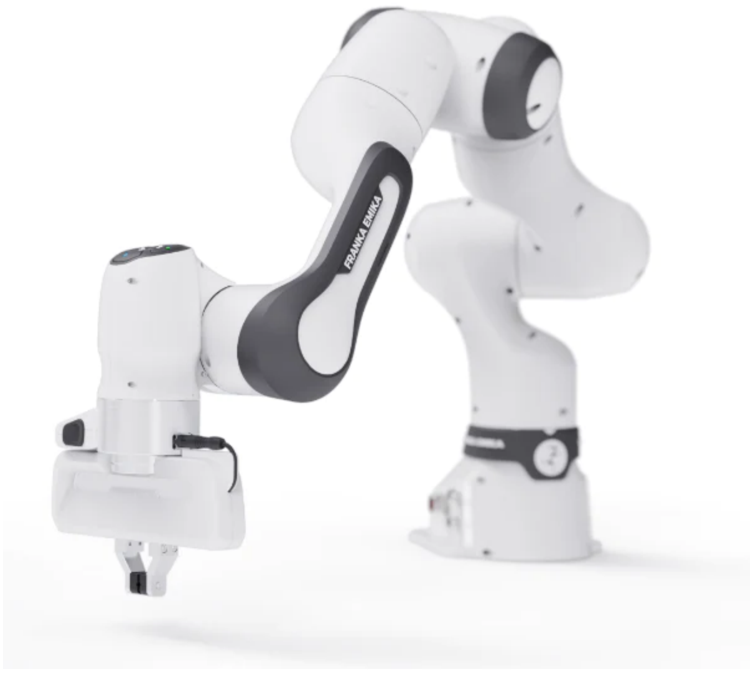


Figure 2.5: Franka Research 3 cobot equiped with its default gripper. force-sensitive cobot designed for cutting-edge AI and robotics research. The robot has 7 degrees of freedom, a payload of 3 kg, and a maximum reach of 855 mm.

Sensors

Autonomous mobile manipulators are highly dependent on sensors to perceive and understand their environment. There are two main categories of sensors that are used in robotics, namely proprioceptive and exteroceptive sensors.

Proprioceptive sensors are a type of sensor that measures values that are internal to the robot. These sensors are commonly used in robots and play a critical role in ensuring their precise control and operation. For example, wheel/motor sensors measure the rotation of a robot's wheels or motors, allowing it to estimate the distance traveled and changes in orientation. Robotic arms also rely on proprioceptive sensors such as joint angle sensors and torque sensors, which measure the angles between the links of a robotic arm and the torque applied by a motor at a joint, respectively.

On the other hand, exteroceptive sensors acquire information from the robot's environment. These sensors measure external environmental factors such as distance and light intensity. Active ranging sensors emit a signal and measure the time it takes for the signal to bounce back after hitting an object. These sensors can be used to determine the distance to objects in the environment. Vision-based sensors use cameras to capture images of the environment and can be used for object recognition, navigation, and obstacle avoidance. [5, 19]

In a laboratory setting with a mobile manipulator, various proprioceptive and exteroceptive sensors should be utilized to enable manipulation and autonomous navigation. Encoders, Inertial Measurement Units (IMUs) and torque sensors are essential for this purpose. Additionally, a vision system or laser scanning is necessary to navigate the environment and avoid obstacles. In certain cases, it may be beneficial for the robot to operate in darkness, as in the work Mobile Robotic Chemist from Toronto University [3], where laser scanning and touch feedback were used instead of a vision system for those reasons. For increased safety and hazard detection, the robot can be equipped with various sensors, such as temperature, humidity, gas and chemical sensors.

2.1.2 Software

To effectively operate an autonomous mobile manipulator, the software structure must be carefully designed and divided into multiple steps. Firstly, the environment must be perceived using information gathered from sensors. Secondly, the path and motion planning step utilizes the perceived information to plan the movement of both the mobile base and manipulator. Finally, the control system directs and manages the behavior of the devices in the system. [25]

Path and Motion Planing

Motion planning algorithms are a critical component of autonomous robotic system, as they enable robots to navigate through their workspace in a safe and efficient manner. These algorithms are used to find the path between a starting point and an endpoint, which allows the robot to move from one location to another.

Path planning is a preliminary step in motion planning, and it involves finding the shortest distance or time between two points at a topological level. However, motion planning goes a step further by taking into account the dynamics of the environment and the robot itself. This includes factors such as the robot's kinematics, velocities, and poses, as well as the presence of any dynamic objects in the environment.

By considering these variables, motion planning generates interactive trajectories that allow the robot to safely and effectively interact with its surroundings. The goal of motion planning is to achieve long-term optimal strategies by utilizing short-term optimal or suboptimal strategies to respond to changing environmental conditions. [26]

In general, path planning algorithms can be categorized into Graph-search, sampling-based and interpolating curve algorithms.

Graph-search Algorithms involve searching for a path through a graph that represents the robot's configuration space, taking into account its geometry and any obstacles or other constraints in the environment. This configuration space is often represented as a discrete graph, where each node in the graph represents a valid configuration of the robot, and each edge represents a valid motion that the robot can make from one configuration to another. The search algorithms used for motion planning operate on this graph by constructing a search tree that explores the possible paths from the initial configuration to the goal configuration, with the goal of finding a shortest path on the graph. Examples of graph-search algorithms include Dijkstra's algorithm and A* algorithm. [26]

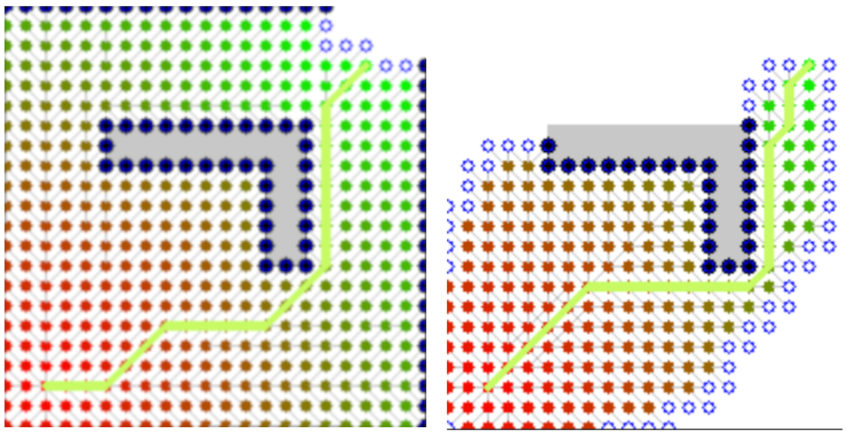


Figure 2.6: Search performance comparison of Dijkstra (left) and A* (right) algorithms in a 2D environment. The obstacle is shown as a gray shape. Red and green circles represent nodes in the closed set, that is nodes that have been evaluated, whereas blue circles represent the set of nodes that have yet to be evaluated. [6]

Sampling-based Algorithms use random configurations to explore the configuration space instead of discretizing it. Local planners connect these random configurations, and then check if there is a feasible path between them that avoids obstacles. This process results in either a probabilistic roadmap (PRM) or a rapidly-exploring random tree (RRT) that covers the configuration space.

Both PRM and RRT are probabilistic in nature, meaning they do not guarantee finding the optimal path or even a feasible solution. However, they are very effective in high-dimensional or complex environments, where exact solutions are often computationally intractable.[26]

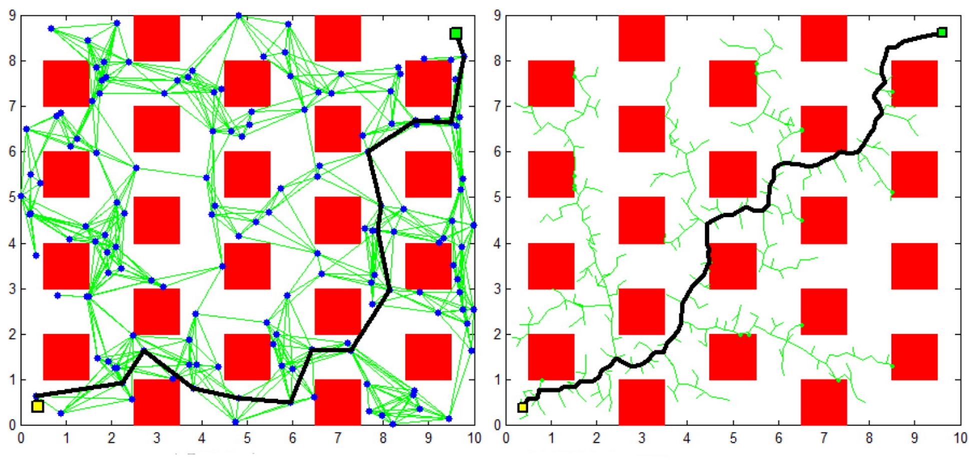


Figure 2.7: Performance comparison of PRM (left) and RRT (right) algorithms in a 2D environment. The obstacle is shown as a red square. Yellow square represents the starting point and green squere the goal point. [9]

Interpolating Curve Algorithms do not use graphs or samples, but rather generate smooth curves that interpolate between waypoints. The waypoints are either given by the user or computed by other methods. The curves are designed to satisfy certain criteria such as curvature, clearance and optimality. The problem then reduces to finding an interpolating curve that passes through all waypoints without colliding with obstacles. Examples of interpolating curve algorithms include B-splines, Bezier curves and Dubins curves.[26]

Motion Planning for Mobile Manipulators
Mobile manipulators pose unique challenges for planning algorithms due to their kinematic redundancy and dynamic complexity. Kinematic redundancy occurs when a system has more degrees of freedom than necessary to perform a given task. With mobile manipulators, this is because of the combination of a mobile base and robot manipulator, which have more degrees of freedom when combined than they do individually.

This redundancy leads to the challenge of determining the most efficient and effective way to achieve a goal. There are multiple ways to achieve the same objective, and the planning algorithm must choose the optimal path. Furthermore, dynamic complexity arises from the different dynamic behavior of the mobile base and the manipulator. This makes it challenging to model and control the system's motion accurately.

Due to this specific challenges of mobile manipulators the planing algorithms are usually implemented in one of two ways and therefore can be categorized into Separate planing and Combined (Whole-body) planing which is based on consideration of planning algorithms of the differences between the robot base and manipulator arm.

Separate Planing Approach divides the complex task that has to be achieved by mobile manipulator into the sequence of sub-tasks and the planing is carried out for each sub-task separately and therefore separates the planning for the mobile base and manipulator arm.

When dividing a task into sub-tasks, the placement of the mobile base can play a crucial role. Poor placement can make the overall task goal unreachable and result in suboptimal solution paths. Since the goal state of one sub-task heavily affects the planning of the next task, it could prevent the generation of a feasible solution, thus resulting in the need for re-planning of the previous task. This can lead to locally optimal but globally suboptimal paths or unsuccessful planning queries.

Whole-body Motion Planning is an alternative approach that considers the robotic system as a single entity with numerous degrees of freedom. However, this approach can be computationally expensive compared to planning each body separately. Specifically, calculating the free space representation and finding a path through it can be highly demanding. The free space denotes the area where the robot can move without colliding with obstacles, and if the obstacles are dynamic, the free space representation needs frequent updating, further increasing computational cost. To avoid computing the free space representation, an alternative is to use graph-search algorithms to create a graph through the free space and identify feasible paths, which is a less expensive approach.

Approaching motion planning as an optimization problem of finding the trajectory is another viable method. However, this approach can be challenging and costly to implement in real-world applications due to the requirement of large memory for storing paths and complex cost functions. Despite these challenges, optimization-based methods can provide precise and accurate paths.

Another popular approach is to use sampling-based algorithms because of their effectiveness in high-dimensional and complex environments. However, generated paths may not be smooth, which can result in less accurate trajectories. Despite this, sampling-based methods are still beneficial due to their ability to handle obstacles of inconsistent shape and size, and their probabilistic completeness property, meaning that given enough time, they can find a feasible path with high probability. [16]

Motion Planing Frameworks

Developing robotic applications can be complex and time-consuming, particularly when it comes to creating algorithms for motion planning. However, there are existing tools and frameworks that can simplify the process. Robotic Operating System 2 (ROS2) [10] is a widely used open-source framework for robotics development. It offers a variety of libraries and tools that make it easier to build robotic applications.

Two such libraries are Navigation Stack2 (Nav2) [11] and MoveIt 2[5]. Navigation Stack provides algorithms for autonomous navigation of mobile robots, while MoveIt provides tools for motion planning for robotic arms. Both libraries are built on top of ROS and are well-established in the robotics community.

By using these libraries, researchers and developers can implement state-of-the-art motion planning algorithms without having to write complex code from scratch

Control System

A control system is a system that manages, commands, directs, or regulates the behavior of other devices or systems using control loops. Control systems can be classified into two types: open-loop and closed-loop. Open-loop control systems do not use feedback and act only on the basis of input signals. Closed-loop control systems use feedback to compare the output with the desired output and adjust their actions accordingly to minimize an error.

Mobile manipulators are controlled in one of three ways: manual, semi autonomous, or fully autonomous control. Manual control, which has been the standard approach, involves direct control of the systen through means such as joysticks. However, this method requires extensive training to achieve competence.

Semi-autonomous control, on the other hand, combines the advantages of both manual and autonomous control. This approach allows for the machine to operate with a degree of autonomy while still being under human supervision. For instance, a mobile manipulator may be programmed to move to a certain location autonomously, but the operator can intervene and modify the trajectory if needed.

Autonomous mobile manipulators are capable of operating without human intervention or supervision, perceiving their environment, planning actions, executing tasks, and adapting to changes or uncertainties. This capability can lead to more efficient and intelligent operations.

There are different approaches for achieving autonomous control of mobile manipulator systems, including behavior-based control, model-based control, and learning-based control. Behavior-based control relies on a set of predefined behaviors that are triggered by sensory inputs or internal states, which can be prioritized, blended, or coordinated to achieve complex tasks. Model-based control uses a mathematical model of the robot and its environment to predict the outcomes of actions and select the optimal ones based on criteria such as cost, reward, or safety. Learning-based control utilizes data-driven techniques like reinforcement learning to learn from experience and improve performance over time. [25]

[5]https://moveit.ros.org

Perception

Mobile manipulators rely on information from multiple sensors to perceive their environment. However, the inherent imperfections in each sensor can result in uncertainty in the information they provide. Thus, effective strategies are necessary to manage the sensory information despite these uncertainties. Two common strategies are used: utilizing the raw measurements from each sensor to inform the robot's actions, or extracting features from one or more sensor readings to interpret the scene.

Feature Extraction is a process of identifying and extracting important or relevant information from sensor data. This can involve extracting features based on range data, or visual appearance-based feature extraction. The objective of feature extraction is to reduce the amount of data that needs to be processed while retaining important information that can be used for tasks such as localization and mapping.

Features are identifiable structures or patterns in the environment that can be extracted from measurements and mathematically represented. Low-level features refer to geometric primitives such as lines, circles, or polygons, which provide an abstraction of raw data and increase the distinctiveness of each feature while reducing the data volume. High-level features, on the other hand, refer to objects such as edges, doors, or tables, which provide maximum abstraction from raw data and reduce the data volume while providing highly distinctive resulting features.

Feature Extraction Based on Range Data involves the identification and extraction of crucial information from sensor data that measures the distance between the mobile manipulator and its surroundings. This strategy employs sensors such as laser or ultrasonic. The extracted features can help represent the environment in a more condensed and informative manner, which can assist with tasks such as localization and mapping. For instance, a laser range finder can be utilized to extract features such as lines or corners in the environment, which can be used to represent walls or other objects.

Visual Appearance-based Feature Extraction involves the identification and extraction of significant information from visual sensor data. This process includes the use of techniques such as color or texture analysis to extract features that represent the visual appearance of objects in the environment. The extracted features can aid the mobile manipulator in recognizing and identifying objects or landmarks in its surroundings. [19]

2.2 Related works

The purpose of this section is to provide an analysis of the implementation of autonomous mobile manipulators in various industries and their potential applications in MAPs. The aim is to present an overview of the hardware and software components used in the implementation of such systems, with a focus on their versatility and potential. Another objective is to guide the final problem formulation of this project by examining the use cases of autonomous mobile manipulators and their potential in solving various tasks, such as material handling, assembly, and inspection. This analysis will provide valuable insights into the potential applications of autonomous mobile manipulators in different industries, including manufacturing, healthcare, and logistics, and their ability to operate in diverse environments, both indoors and outdoors.

2.2.1 Little Helper 8

Aalborg University researchers have developed a series of autonomous mobile manipulators called Little Helpers (LH) [20] to improve the flexibility, speed, and efficiency of production and logistics handling. Unlike most static industrial manipulators that are limited by their workspace, LH models were developed with specific aims, including logistics and multiple part feeding, assembly and machine tending, and a dual-arm robot co-worker. The latest model, LH8, was designed with an omnidirectional mobile base, allowing lateral movement, in contrast to all previous versions, which had differential drive mobile platforms. Adding an additional degree of freedom (DOF) to the system enables LH8 to navigate shorter paths than differential drive mobile platforms, as shown in figure 2.8. To perceive its environment, LH8 uses two LIDAR sensors.

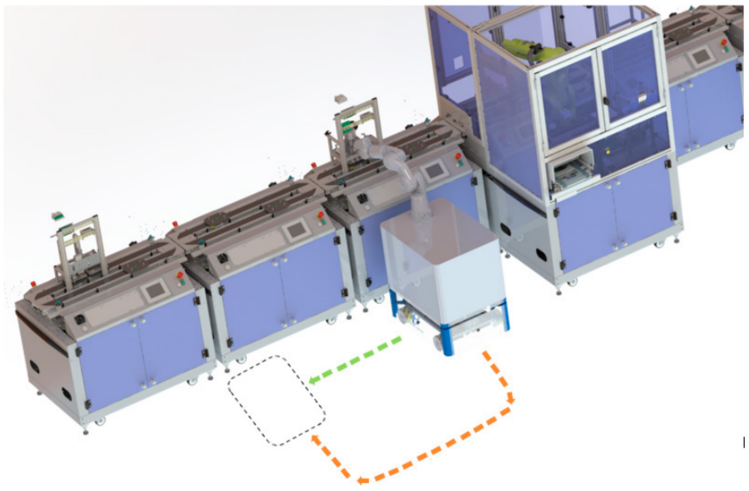


Figure 2.8: The figure illustrates the path of an omnidirectional LH8 robot, denoted by the green line, in contrast to the path of a differential drive robot, represented by the red line.[20]

The navigation package was programmed in ROS, which implemented global and local planners. The global planner utilizes a graph-search-based A* algorithm for long-term planning, from the initial pose to the goal pose. The local planner serves as an online local trajectory planner for omni-drive, which optimizes the initial trajectory generated by the global planner while handling static and dynamic obstacle avoidance. To localize the robot in the map, this work implements adaptive Monte Carlo localization (AMCL). The tests were conducted in both the Gazebo simulation tool and real-life scenarios, wherein the robot was able to avoid obstacles and navigate from the initial position to the goal position with a significant margin of error of 0.33m, attributed to the use of cheap LIDAR sensors. However, no tests were performed for the pick and place task as the robot arm was not mounted on the mobile base.

2.2.2 A Mobile Robotic Chemist

This thesis focuses on the utilization of autonomous mobile manipulators in Material Acceleration Platforms (MAPs), which are designed to accelerate the discovery of new materials through high-throughput experimentation. A noteworthy example of such a system is the implementation of a KUKA mobile robot mounted on a KUKA Mobile Platform base by researchers from the University of Liverpool [3]. This robot was deployed in a laboratory environment and could perform complex manipulations comparable to those performed by human researchers. The robot could autonomously move around the laboratory, select and weigh reagents, mix them together, apply heat, and measure the gas evolved. Moreover, it utilized artificial intelligence (AI) to analyze the data and plan the next experiments using a Bayesian search algorithm.

The primary objective of this research was to search for improved photocatalysts for hydrogen production from water. The robot carried out 688 experiments within a ten-variable experimental space over eight days, achieving this objective. To navigate its environment, the robot employed a cloud of possible positions and updated its position by matching the output of its laser scanners with the map for each position in the cloud. The robot's position was determined by x, y, and θ (its orientation angle). Furthermore, a touch-sensitive 6-point calibration method was utilized to enhance precision, allowing the robot to operate instruments and carry out delicate manipulations such as vial placements with a level of precision comparable to a human operator.

The researchers implemented a Bayesian search algorithm to guide the robot in its search for improved photocatalysts for hydrogen production from water. The objective function was finding the optimal set of concentrations in a multicomponent mixture for photocatalytic hydrogen generation. This enabled the robot to efficiently explore a large experimental space and identify promising solutions.

This study demonstrated the potential and benefits of autonomous mobile manipulators in MAPs and laboratory environments, as well as the challenges and opportunities for future work on mobile robotic chemists, such as improving their safety, reliability, robustness, scalability, and collaboration. Overall, this research showcases the tremendous potential for autonomous mobile manipulators to revolutionize high-throughput experimentation and expedite the discovery of new materials.

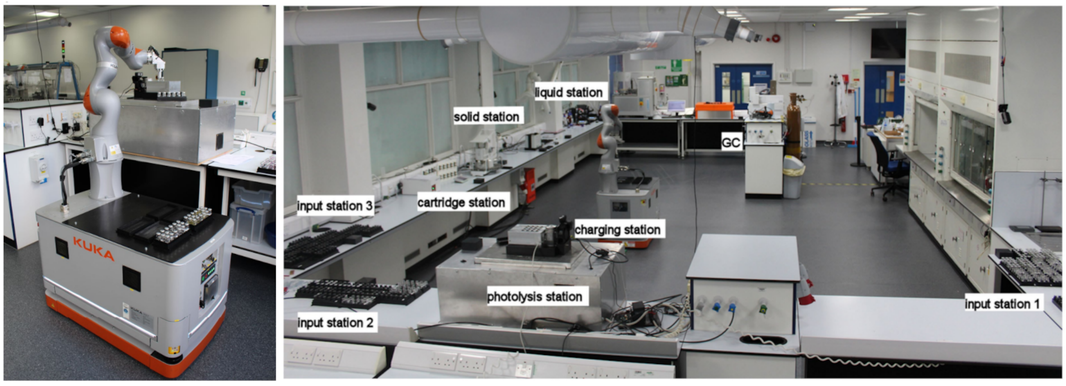


Figure 2.9: KUKA Mobile Robot mounted on a KUKA Mobile Platform base loading samples into the photolysis station (left). Laboratory environment used for performing experiments.(right) [3]

2.2.3 Autonomous Mobile Manipulator for Construction-based Tasks

Autonomous mobile manipulators are increasingly being utilized in various domains, and the work of Basiri et al.[1] provides an interesting use case for their application in outdoor structure building with heterogeneous brick patterns. Specifically, the researchers implemented an autonomous mobile manipulator consisting of a refurbished ATRV-Jr mobile base and a UR5e robotic arm, shown in figure 2.10. To enable localization and navigation, the system is equipped with several sensors, including a GPS antenna and receiver, two IMUs, a laser scanner, and an Intel RealSense camera with RGB and depth sensors positioned near the end-effector.

In order to achieve autonomous behavior, the work employed a state-machine where predefined states were autonomously switched based on sensor inputs and mission status. To enhance the localization performance, an Extended Kalman Filter was employed that combined odometry information, GPS and IMU measurements to output a pose estimation for the robot to safely move around the environment. The robot further utilizes a laser scanner to precisely position itself near the pile of bricks or the wall, enabling it to reach, grasp, and manipulate bricks.

To plan its motion, the work utilized Dijkstra's Algorithm based Global planner and Dynamic Window Approach (DWA) Local Planner based on ROS navigation stack. This approach benefits from costmaps to represent obstacles and build an occupancy grid representation of the environment, enabling the robot to plan its path efficiently and avoid obstacles.

Overall, this work demonstrates the potential of autonomous mobile manipulators for outdoor structure building tasks, highlighting the importance of localization and motion planning techniques for their successful operation. Future work can build upon this foundation by addressing challenges such as improving the robustness, scalability, and reliability of such systems.

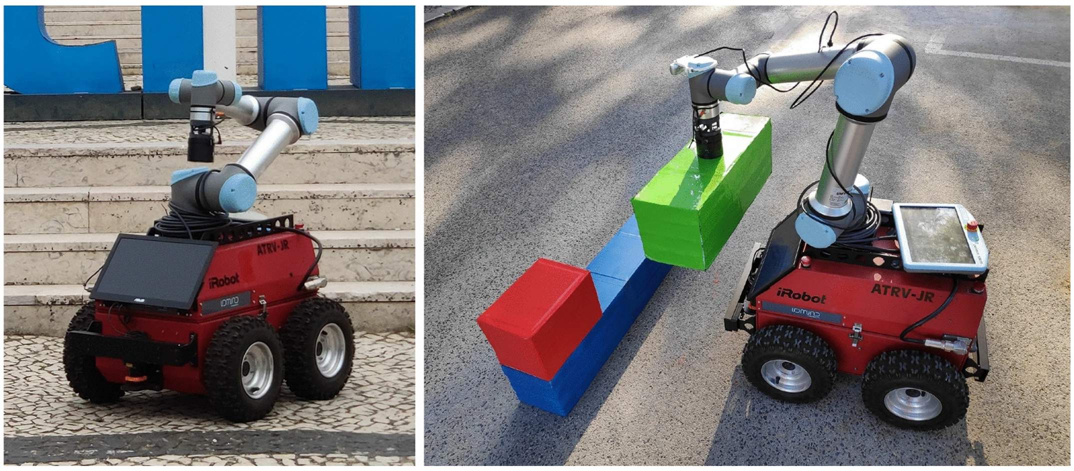


Figure 2.10: The mobile manipulator assembled from refurbished ATRV-Jr mobile base and a six-degree-of-freedom UR5e robotic arm.[1]

2.2.4 Whole-Body Mobile Manipulation

The conventional approach to mobile manipulators involves separately planning the motions of the mobile base and robot arm. However, Mittal et al.[12] propose a novel approach to whole-body motion planning for manipulating articulated objects in novel scenes. Their paper introduces a two-level planning hierarchy: object-centric and agent-centric. The former provides proposals for interacting with the object without considering scene information or the agent embodiment, while the latter ensures safe execution of the proposed plans by the robot, accounting for its dynamics and environment.

To benchmark the feasibility of successful articulated object manipulation in unknown environments, the authors explore three different planners for mobile manipulators that use map information for environment collision avoidance. These include: i) sampling-based planning for the base with inverse kinematics (IK) for the arm, ii) whole-body control using IK, and iii) whole-body control via model predictive control (MPC). They observe that the MPC-based solution performs significantly better than IK-based solutions in terms of success rate and execution time.

The authors conducted experiments on three different kitchen layouts, shown in figure 2.11, using assets from the PartNet-Mobility dataset. Each layout contains everyday objects found in modern kitchens, such as tables, refrigerators, and microwaves. The task was for a kitchen assistant robot to operate human-scale objects such as cabinets and ovens in unmapped environments with dynamic obstacles.

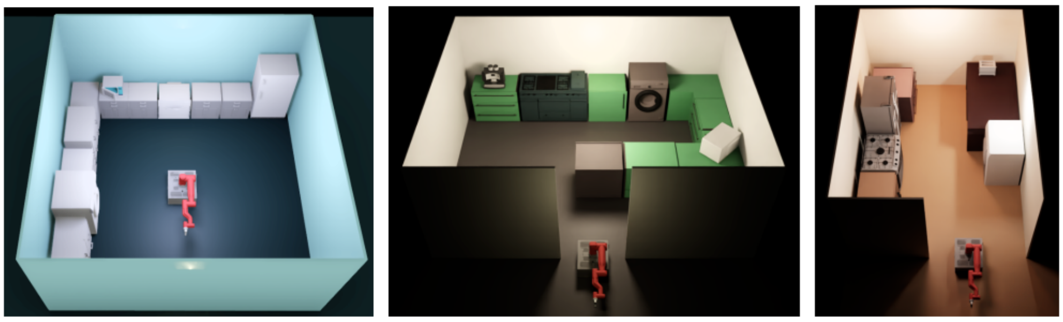


Figure 2.11: The image represents three distinct kitchen settings used in the testing process.[12]

The simulation tool used for experiments was NVIDIA Isaac Sim, which utilizes PhysX for stable, fast, and realistic physics simulations, as well as multi-GPU ray tracing for photorealism. The platform used in the simulation was Mabi-Mobile, which consists of a differential drive base with four supporting castors and a 6-DoF manipulator. Hardware tests were also performed on an ANYmal-D platform equipped with a 6-DoF torque controllable robotic arm. The implemented robots are shown in figure 2.12.

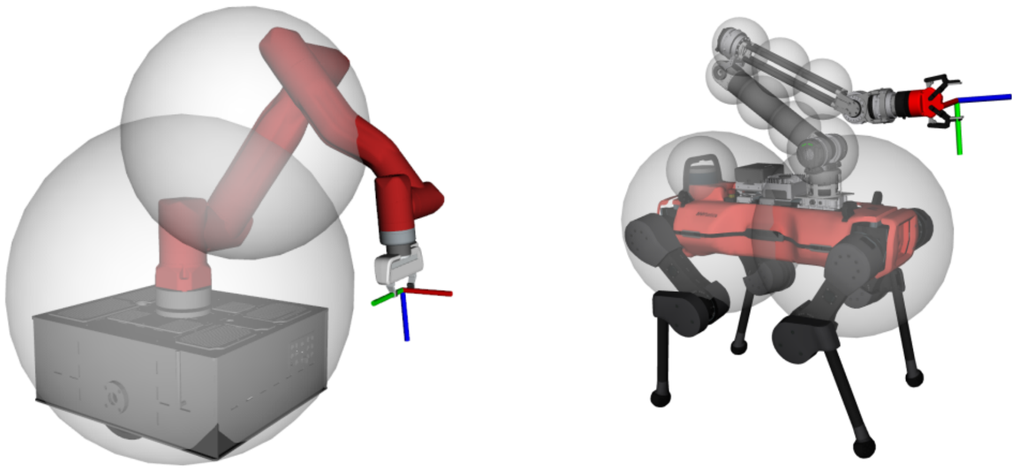


Figure 2.12: The iamge depicts implemented mobile manipulators in the work, specifically the Mobi-Mobile wheeled robot (left) and the ANYmal-D platform with a robotic arm (right). The collision mesh for both robots is approximated using spheres.[12]

The results of the experiments showed that the proposed method outperformed other state-of-the-art methods in terms of success rate and execution time. Moreover, the authors demonstrated that their method can handle complex kitchen settings and ensure safe execution for both wheel-based and legged mobile manipulators.

2.3 Summary of Analysis

Autonomous mobile manipulators have emerged as a promising technology for a wide range of applications, including material handling, assembly, and inspection. Researchers have implemented such systems in various industries and have demonstrated their potential in both indoor and outdoor tasks. Autonomous mobile manipulators have also been implemented in MAPs to accelerate the discovery of new materials through high-throughput experimentation. These systems employ artificial intelligence (AI) to analyze data and plan experiments efficiently. Moreover, autonomous mobile manipulators have demonstrated success in outdoor environments, particularly in construction-based tasks. Benefits of whole-body motion planning over traditional separate body motion planning approach has also been examined. Overall, findings from related works showcase the great potential of autonomous mobile manipulators to advance high-throughput experimentation and expedite the discovery of new materials.

Chapter 3

Problem Formulation

This chapter presents the final problem formulation that will guide the design of this thesis's solution, which is based on the previous chapter's analysis. Furthermore, the project's task and goals will be defined in order to guide the development and evaluation of the proposed solution and to more precisely specify the project's scope.

3.1 Final Problem Formulation

The problem analysis investigated the design of an autonomous mobile manipulator system from both the hardware and software aspects. As a result of this examination, it was discovered that these robots have great potential in a variety of industries due to their ability to provide high levels of flexibility and efficiency. However, when it comes to motion planning and system control, the dynamic complexity of these robots presents a significant challenge.

Additionally, the problem analysis investigated some of the existing works in this field in order to gain a more comprehensive understanding of the current research state as well as the practical implementation details and potential applications of these robots.

As a result of this finding the final problem formulation is defined as:

Establish a foundation for simulating an omnidirectional mobile manipulator within a laboratory setting to support future research on CAPeX.

3.2 Task Definition

Drawing inspiration from Liverpool University's Mobile Robotic Chemist project [3], the task of the mobile manipulator is to navigate to the handspace gas chromatography station. This specific station is where the gas phase is analyzed for hydrogen after photolysis, which marks the final phase of the experiment. The mobile manipulator's task at this station is to retrieve the completed rack of vials and transport them to the designated storage area.

The simulated environment setup used in this work is shown in figure 3.1, the assets used in the creating of this environemnt are downloaded from GrabCad[1] an online platform for sharing and downloading 3D CAD models.

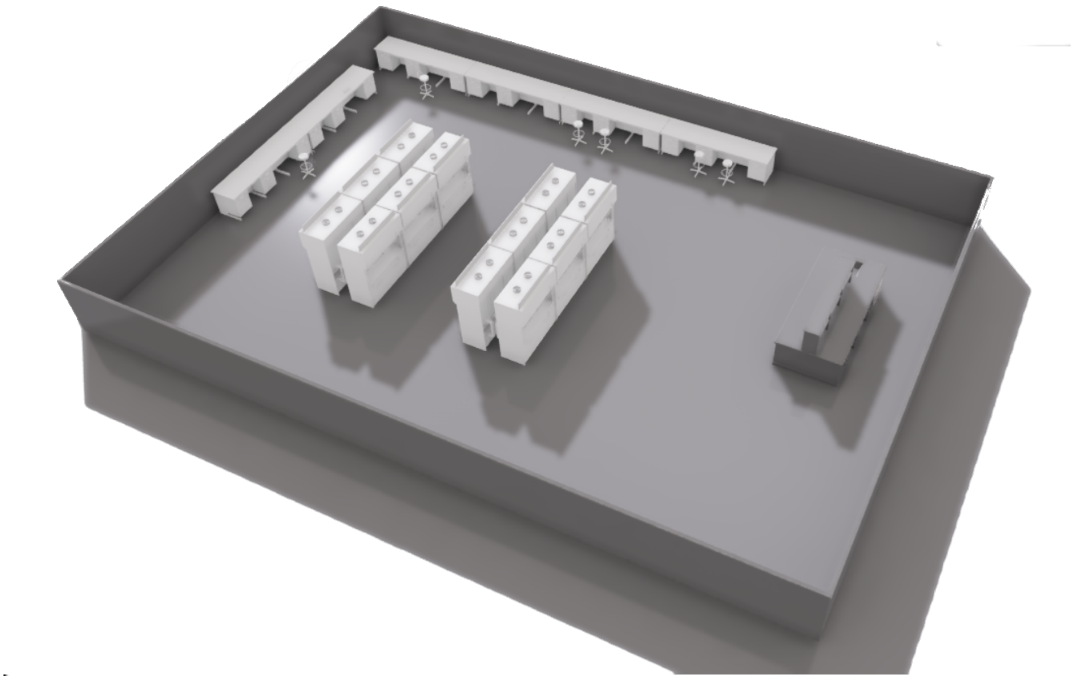


Figure 3.1: The simulated laboratory environment used in this work.

3.3 Project Objectives

Objective 1: Integrate the Summit XL mobile platform with the Franka Emika Panda robot arm to create a mobile manipulator robotic system
This objective involves assembling a 3D model of the mobile manipulator by integrating existing models of the mobile base and robot arm.

Objective 2: Create a photo-realistic simulation of the robotic system using the Nvidia Isaac Sim simulation tool and incorporating ROS2 functionality for integration with Isaac Sim
This objective involves creating a comprehensive simulation environment that includes robotic system, sensors, a control system, and motion planning capabilities.

[1]https://grabcad.com/

The simulation will be based on the ROS2 framework and will demonstrate the execution of a set task.

Objective 3: Create a simulated laboratory environment in which the aforementioned task will be executed
This objective involves creating a an environment which resembles the laboratory setting with at least two stations one for the picking location of the rack with flasks and one station for the deposit of the completed racks.

Objective 4: Perform testing of the proposed mobile manipulator system
This objective involves the testing of the implemented systems functionality both on low-level and high-level. Testing of low-level functionalities include testing the performance of navigation system and robot arm manipulation. High-level functionalities testing should check the capabilities of the system to execute the set task.

Objective 5: Provide a GitHub repository as a documentation for the implemented source code and simulation setup
This objective is set in order to contribute to the CAPeX and support future extensions of this work.

Chapter 4

Solution Architecture

This chapter will define the design of the proposed solution for the mobile manipulator system by defining the hardware components and software developer tools used for navigation, perception, motion planning, and control. In addition, the simulation tool that will be employed will be determined.

4.1 Robot Specifications

This project proposes a mobile manipulator system comprising the Robotnik's Summit XL[1] mobile platform and the Franka Emika Panda[2] robot arm mounted on top.

4.1.1 Mobile Platform

The Summit XL is a versatile mobile robot designed to operate in both indoor and outdoor environments. With modular wheels it comes with two options as shown in figure 4.1. Rubber wheels with skid steering kinematics-configuration and mecanum wheels for an omnidirectional traction system. For this project, the proposed system will utilize the mecanum wheel configuration, leveraging the benefits of omnidirectional movement. This configuration is particularly advantageous in laboratory settings, offering superior maneuverability within indoor environments characterized by flat surfaces, which are well-suited for mecanum wheels.

Additionally, the Summit XL is equipped with 500 W brushless servomotors in each wheel, which are equipped with encoders to accurately detect the rotation speed of the motors. The robot's dimensions are 720 x 614 x 416 mm, with a weight

[1]https://robotnik.eu/products/mobile-robots/summit-xl-en-2/

[2]https://www.franka.de

of 65 kg. It has a payload capacity of 50 kg and can reach a maximum speed of 3 m/s.

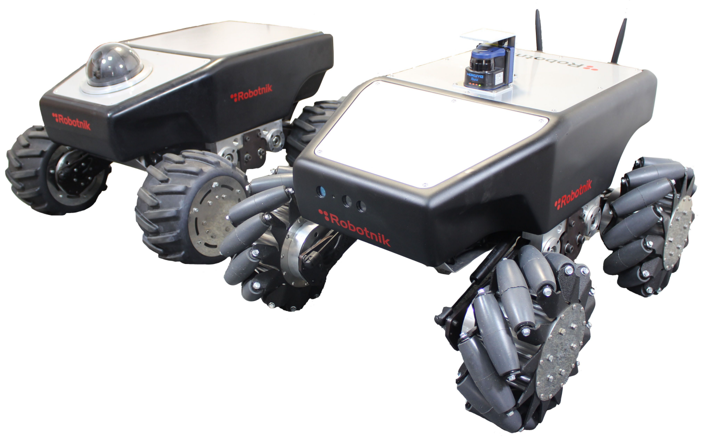


Figure 4.1: Summit XL mobile platform with two different wheel configurations. Rubber wheel configuration (left), and Mecanum wheel configuration (right)

Sensors

To enable autonomous navigation in indoor environments, the mobile platform is equipped with essential sensors. These include a 2D rotational Lidar sensor, an Inertial Measurement Unit (IMU) with integrated gyroscope and accelerometers. These sensors work together to accurately estimate the position and orientation of the mobile robot in space.

Moreover, the mobile platform will be equipped with a front-facing RGB camera, which improves obstacle avoidance capabilities. This camera provides visual information to aid in detecting and navigating around obstacles effectively.

Wheel Configuration

As described in Section 2.1.1, mecanum wheels possess a special design consisting of freely moving rollers mounted around the central hub at a 45-degree angle relative to the wheel's axis. This configuration enables a secondary translation of the mobile base, enabling omnidirectional motion. By applying different velocities to each wheel, distinct movements are generated. A depiction of the velocity and wheel configuration corresponding to specific motions can be seen in figure 4.2.

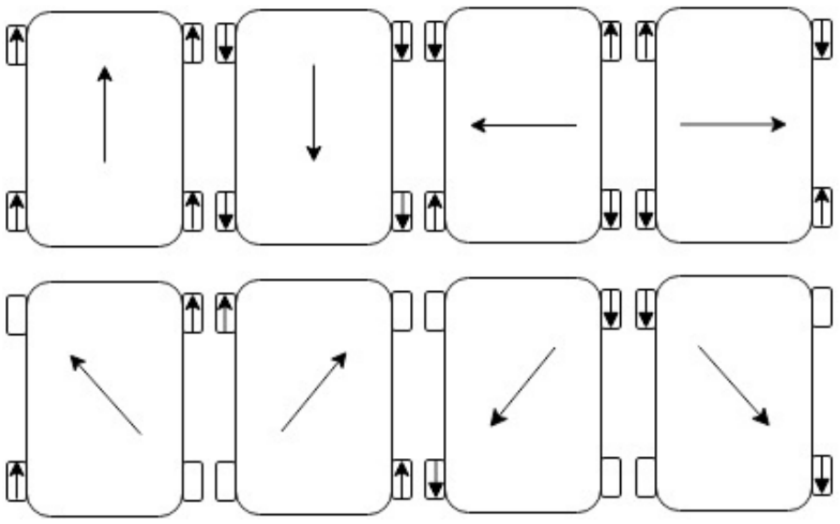


Figure 4.2: Illustration of mobile base motions dependent on wheels' velocities. [20]

4.1.2 Manipulator

The Franka Emika Panda is a seven-degrees-of-freedom (DoF) robotic arm that provides a wide range of motion and dexterity. Having a reach of 855 mm and weighing 18.5 kg. Integrated force/torque sensors are incorporated into each joint. These sensors allow the arm to interact and collaborate with its environment, including human operators, in a safe manner.

The Franka Emika Panda will be mounted on a mobile base to create a mobile manipulator system as part of this project. In addition to the arm itself, additional essential components must be incorporated. The controller, which ensures the communication and control over the Panda's movements. Since the Franka Emika Panda requires a power supply, the incorporation of a power inverter (230V) is required. The power inverter enables the arm to be connected to the power supply, allowing the system to be mobile without interruption.

4.2 Robot Operating System (ROS)

ROS is an open-source framework designed to facilitate the development of robot software. It is a collection of software libraries and tools.

ROS provides a flexible and modular architecture that allows developers to create software components, called nodes. These nodes possess the ability to communicate with one another through a messaging system. By leveraging this mechanism, nodes can exchange various types of data, including sensor readings, control commands, and other relevant information.

ROS packages serve as the fundamental building blocks of the framework, offering a systematic and organized approach to the development and management of robot software. These packages consist of several key components, such as the manifest (package.xml), source code, and launch files.

4.2.1 ROS Topics

ROS topics are the vital messaging system in which the message is moved between the nodes. Nodes can publish messages to topics, and other nodes can subscribe to receive those messages, as seen in figure 4.3. Those messages that are being sent represent the information being communicated through ROS topics and services. They encapsulate specific data types and fields, allowing nodes to understand and interpret the data received or sent.

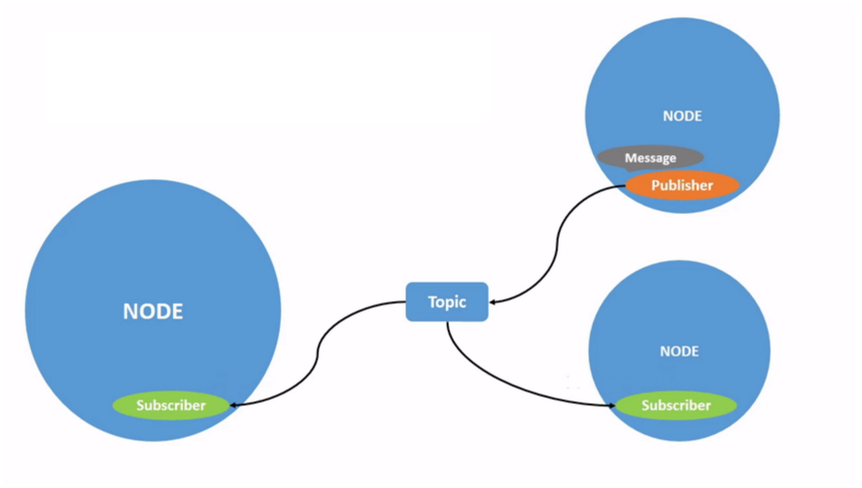


Figure 4.3: The illustration demonstrates the messaging system of ROS topics, showcasing the capability to have multiple publishers and subscribers for a single topic. [10]

4.2.2 ROS Services

Services in ROS represent a call-and-response messaging system, as depicted in figure 4.4. Unlike topics, services offer data only upon client request, rather than providing a continuous data stream.

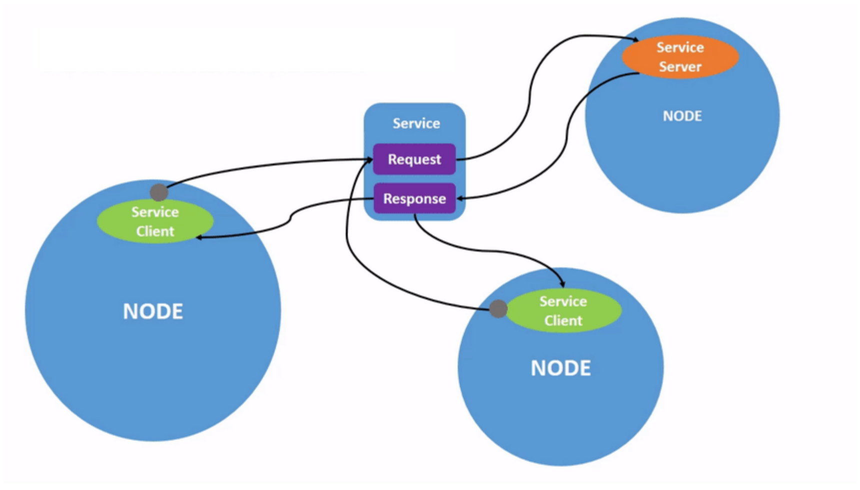


Figure 4.4: The illustration shows the ROS services messaging system, pointing out the ability for multiple clients while only one service server is permitted. [10]

4.2.3 ROS Actions

The last messaging system in ROS is known as Actions, specifically designed to handle long-running tasks efficiently. Actions share similarities with services but offer additional functionality. Unlike services that provide a single response, Actions provide continuous feedback throughout their execution and can be canceled if needed.

Actions follow the action-client model, shown in figure 4.5, and consist of three main components: the goal, feedback, and result. A node initiates an action by sending a goal to an action server node, which accepts and processes the goal. Throughout the execution of the action, the action server provides a continuous stream of feedback data. Finally, upon completion or cancellation, the action server sends a result message to the client node.

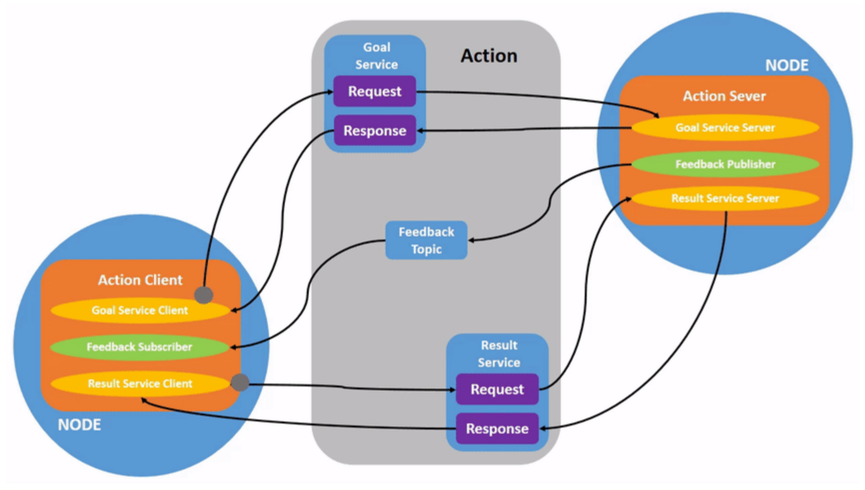


Figure 4.5: Illustration of ROS action messaging system.[10]

4.3 Navigation Stack

Navigation 2 (Nav2) is a set of tools developed for ROS. It serves as a comprehensive framework that enables autonomous robot navigation by integrating various functionalities such as perception, localization, mapping, path planning, and control. Behavior trees are used for the creation of customized and intelligent navigation behaviors by orchestrating multiple independent modular servers.

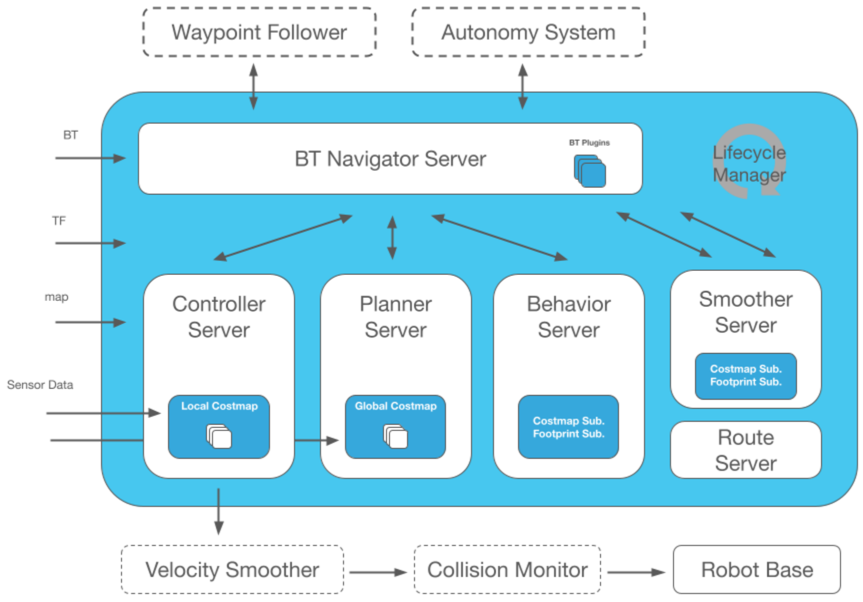


Figure 4.6: Architecture of the Navigation stack.[11]

From the figure 4.6, it is seen that the navigation stack uses four different servers: Controller server, Planner server, Behavior server and Smoother server that improves the planned path.

The Behavior server serves as a means to manage the recovery behavior that is activated when a mobile robot experiences a malfunction or encounters an issue, in addition to facilitating the implementation of various customized behaviors.

The Planner server receives the goal and calculates a feasible path towards it. To accomplish path planning, a variety of plugins can be utilized, each implementing different path planning algorithms. Some examples include the NavFnPlanner, SmacPlannerHybrid, and ThetaStarPlanner.

The Controller server receives high-level navigation commands, such as goals or trajectories, and transforms them into the necessary low-level control commands to achieve the desired robot behavior. These commands are then published as Twist messages on the $/cmd_vel$ topic. The Twist message contains velocities along the x and y axes, as well as the angular velocity for turning around the z-axis.

Similar to the Planner server, the Controller server provides different plugins that implement various algorithms. Some examples of these plugins include the DWB Controller, TEB Controller, and Regulated Pure Pursuit.

Transformations

The transform tree defines relationships between different coordinate systems, accounting for translation, rotation, and relative motion. To publish the transform tree of a robot the TF2 ROS package is commonly used.

In case of Navigation, three essential coordinate frames are used: $base_link$, odom, and map. The $base_link$ represents a coordinate frame set to a fixed position on the robot, typically on its chassis and its center of rotation. The odom coordinate frame is fixed relative to the robot's initial position, and it is primarily used to represent locally consistent distances. Lastly, the map coordinate frame is a world-fixed frame, providing globally consistent distance representations.

For navigation to function, certain transforms must be published:

- `map => odom`
- `odom => base_link`
- `base_link => sensor base frames`

The `map => odom` transform is managed by an Adaptive Monte-Carlo Localizer (AMCL), providing live updates to ensure dynamic values within the robot's transform tree.

The `odom => base_link` transform is typically generated by an odometry system that utilizes sensors like wheel encoders. This transform is typically computed through sensor fusion techniques, using data from multiple odometry sensors.

Furthermore, the `base_link => sensor base frames` transform is a static part of the transformation tree. Nav2 employs this transformation to establish accurate connections between sensor data or other frames of interest and the rest of the robot's components.

4.4 Motion Planning for Robot Arm

Motion planning plays a vital role in robotic systems, enabling them to navigate their workspace safely and efficiently. In the case of a mobile manipulator system, motion planning for the robot arm involves finding a collision-free path from the current configuration to a desired configuration, considering the arm's kinematic constraints and avoiding obstacles in the environment.

For this project, the Moveit2 planning framework will be employed, leveraging its functionalities within the ROS ecosystem. Moveit2 offers comprehensive tools for robot arm motion planning and trajectory execution. It employs various motion planning algorithms, including the aforementioned PRM and RRT algorithms.

In addition, Moveit2 allows for the creation of a collision-free motion by defining collision objects in the scene and incorporating them into the planning process.

Moreover, Moveit2 integrates with the rviz2 visualization tool, enabling real-time monitoring of the planning process. This functionality facilitates the visualization of the robot's state, trajectories, and planning scene. Furthermore, the entire process, including scene creation, motion generation, and trajectory execution, can be performed within the rviz2 graphical interface.

4.5 Simulation Tool

The use of a simulation environment to develop a mobile manipulator system has a number of advantages. It reduces expenses and saves time by removing the need for physical components and accelerating testing and development. Simulations offer a risk-free environment for early identification and correction of design flaws, thereby mitigating potential dangers. Rapid iteration on design parameters and algorithms, performance optimization, and scenario-based evaluation of the system's behavior.

For this project, the NVIDIA Omniverse Isaac Sim [8] is chosen as a simulation tool. It is a simulation toolkit based on the Omniverse platform equipped with the necessary tools and workflows for creating and developing robotic experiments. Isaac Sim leverages the powerful PhysX physics engine for accurately simulating real-world physics interactions and dynamics.

In addition, Isaac Sim offers support for ROS/ROS2 based navigation and manipulation application development. Furthermore, it enables the simulation of a variety of sensors, including Lidar, IMU, and RGB-D, whose parameters are simple to modify and whose data can be published as ROS topics. This capability is essential for the requirements of this project, as it ensures accurate sensor simulation and allows the mobile manipulator system's integration.

Chapter 5

Implementation

This chapter focuses on highlighting the implementation details of the proposed solution. It provides a explanation of how the navigation system is implemented in the Isaac Sim for an omnidirectional mobile robot platform. Additionally, it covers the integration of a robot arm into the system, making it into a mobile manipulator. Furthermore, the chapter explains the implementation of the Moveit2 platform for motion planning and trajectory execution for the robot arm. Finally, an overview of the system as a whole is provided, with a specific emphasis on the communication between different sub-components of the system.

5.1 Robot Description

The description of the mobile platform is obtained from the Robotics GitHub repository[1], which contains all the necessary packages for navigation on the platform in both the real world and the Gazebo simulation. To ensure the mecanum wheels function realistically in a physics-based simulation environment such as Isaac Sim, the robot's description must be modified accordingly.

In addition, the existing model within Isaac Sim will be employed for the Franka Emika Panda robot arm, along with the readily available arm controllers.

5.1.1 Rollers definition

To enable omnidirectional movement, it is crucial to define the rollers and equip each of them with a revolute joint that allows free rotation around its axis, shown in figure 5.1.

In the Universal Robot Description (URDF) file imported into Isaac Sim, the xacro (XML Macro) functionalities are used to achieve the desired task. The im-

[1]https://github.com/RobotnikAutomation/

plementation of these macros is adopted from the GitHub repository[2] . The code employs recursion by calling the macro itself until the specified number of iterations, which corresponds to the number of rollers on the wheel, is reached. This recursive behavior is achieved by a conditional if statement that ensures the loop executes only when the *loop* parameter is non-zero.

The joints' origin positions and orientations are determined based on the following equations:

For side = 1 (right wheels):

- Origin position (x, y, z): $(0, 0.0895cos(th), 0.0895sin(th))$
- Origin orientation (in roll-pitch-yaw): $(\arctan(-\tan(th)\sqrt{2.0}, \frac{\arcsin(-\cos(th))}{\sqrt{2.0}}, \arctan(\sin(th)))$

For side = -1 (left wheels):

- Origin position (x, y, z): $(0, 0.0895\cos(th), 0.0895\sin(th))$
- Origin orientation (in roll-pitch-yaw): $(\arctan(-\tan(th)\sqrt{2.0}, \frac{\arcsin(-\cos(th))}{\sqrt{2.0}}, \arctan(-\sin(th)))$

Where,
$th = \frac{360}{num(loop-1)}$

The *num* parameter represents the number of rollers on a wheel, and $loop - 1$ represents the current roller in a loop.

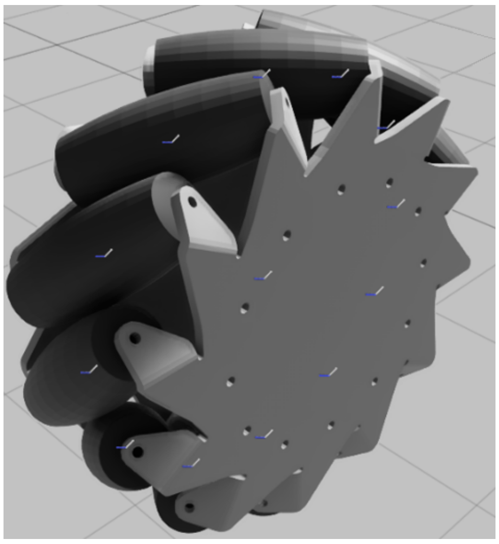


Figure 5.1: Position and orientation of the rollers revolute joint.

[2]https://github.com/DaiGuard/

Furthermore, to enhance omnidirectional movement performance in the simulation, the rollers are represented in the collision mesh as a set of five spheres, as depicted in figure 5.2. This method reduces the number of contact points between the rollers and the ground, leading to improved computational efficiency for collision simulation and overall simulation performance.

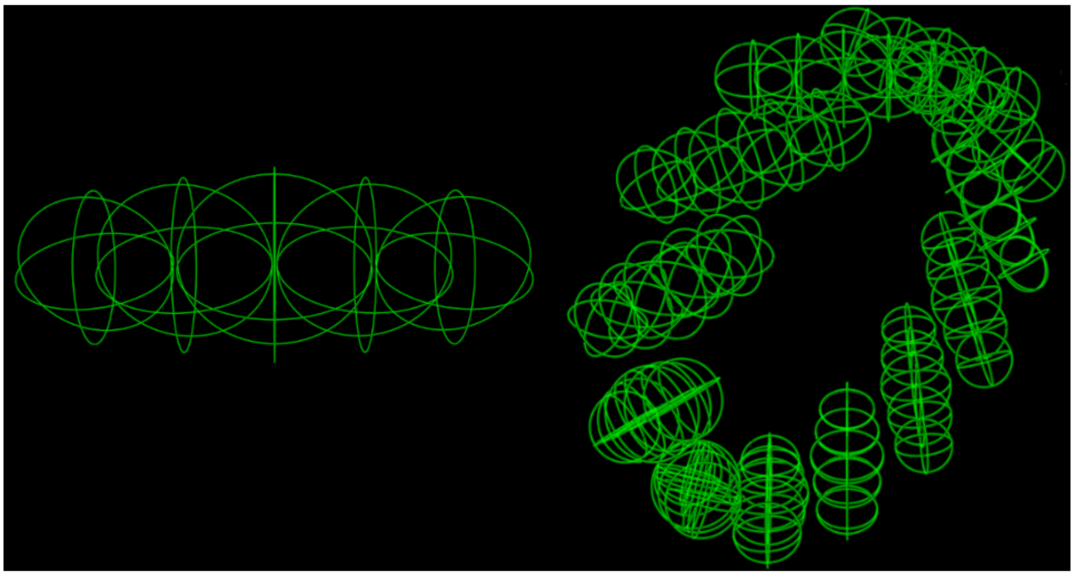


Figure 5.2: Collision mesh of a single roller (left). Collision mesh of all the rollers on a wheel (right).

5.2 Holonomic Controller

The mobile platform's configured model now includes drivable wheel joints and freely rotating rollers, allowing the robot to be easily moved by directly inputting the desired velocity to the actuated joints. However, when it comes to autonomous navigation and utilizing Nav2, which outputs the desired velocity of the robot in the form of a Twist ROS message, a controller must be implemented to convert these messages into the velocity of each wheel.

Isaac Sim offers the solution by leveraging its visual scripting framework called OmniGraph. This framework enables the implementation of action graphs, which facilitate event-driven behaviors and can be used in conjunction with ROS communication.

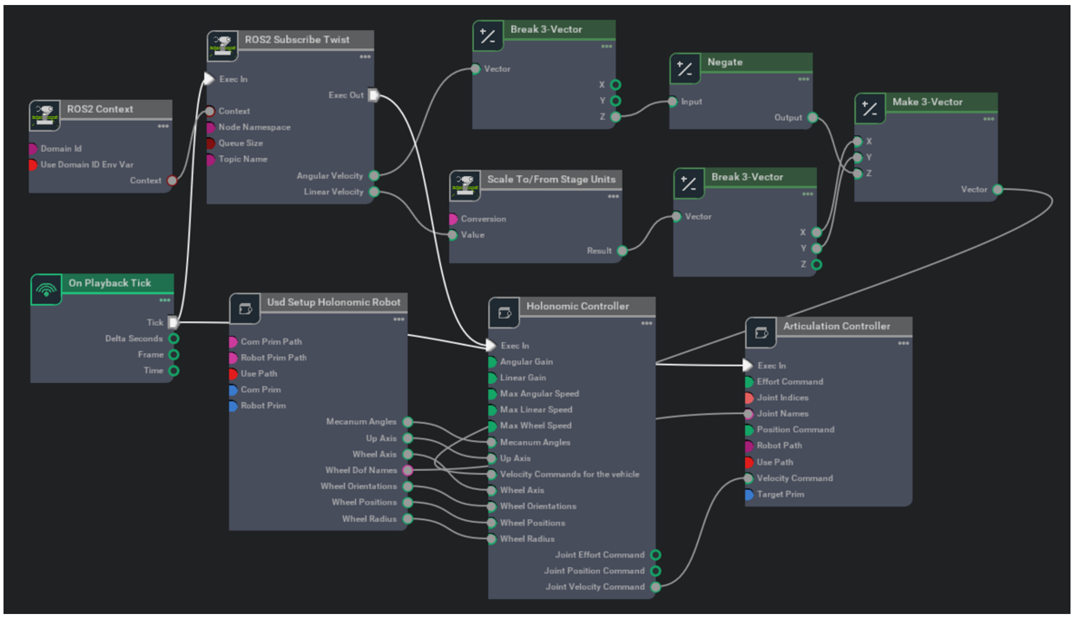


Figure 5.3: The action graph is used to subscribe to the `cmd_vel` topic, configure the holonomic controller, and apply the calculated wheel velocity command to the articulated mobile robot.

Figure 5.3 illustrates the configuration of the action graph used for subscribing to the `cmd_vel` topic, where nav2 publishes desired velocities for robot navigation. The ROS2 subscriber node is responsible for subscribing to the `cmd_vel` topic and forwarding the values to the holonomic controller.

To facilitate the creation of the robot's mathematical model, the USD Setup Holonomic Robot Node is utilized. This node requires the names of the actuated wheel joints, which have been previously modified with additional attributes: mecanuumwheelangle and mecanuumwheelradius. These attributes store information about the mecanum wheel angles and wheel radii. The node sends this information to the holonomic controller node, along with the computed positions of the wheels relative to the vehicle's center of mass and the wheel orientations represented as quaternions in relation to the vehicle's center of mass frame.

Equipped with all the necessary information to construct the mathematical model of the robot and the desired vehicle velocity from Nav2, the holonomic controller node calculates the velocities for each individual wheel of the robot. These velocities are then passed to the Articulation Controller Node, along with the drive joint names and indices, which apply the target velocities to the corresponding joints in the simulation.

5.2.1 Inverse Kinematics

To accurately represent the behavior of an omnidirectional mobile platform, it is essential to develop a mathematical model that includes its inverse kinematics. Taheri et al. [23] formulated the inverse kinematics of such platform as follows:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & 1 & -(l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ w_z \end{bmatrix} \tag{5.1}$$

where,

- $w_1, w_2, w_3,$ and w_4 represent the angular velocities of the four Mecanum wheels.
- r is the wheel radius
- l_x and l_y are half the distances between the front wheels and between the front and rear wheels, respectively
- $v_x, v_y,$ and w_z are the linear velocities of the robot in the x and y directions and its angular velocity about the z-axis, respectively.

To solve the Inverse Kinematics of the four mecanum wheeled robot, that is, to calculate the velocity of each wheel given the desired velocity of the robot, the holonomic controller node formulates the problem as a quadratic optimization problem and uses the Operator Splitting Quadratic Program (OSQP) solver [21] to solve it.

The optimization problem is formulated as:

$$\begin{aligned} &minimize \quad \frac{1}{2} x^T P x, \\ &subject\ to \quad l \leq Ax \leq u \end{aligned}$$

where,

- **x** is the optimization variable that corresponds to the wheel speeds. It represents the vector of wheel speed values that are being optimized to achieve the desired vehicle motion.
- **P** is the matrix that represents the quadratic cost matrix in the optimization problem. It is constructed as a diagonal matrix using the wheel radius values normalized by their Euclidean norm.

- **A** is the matrix that represents the linear and angular velocity constraints of the mecanum wheel system in the optimization problem. Which is based on position, direction, radius and velocity of the wheel.
- **l** and **u** are vectors that represent the lower and upper bound vectors. These bounds are used to define the constraints on the wheel speeds, ensuring they satisfy the specified limitations on linear velocities and angular velocities of the joints.

5.3 Occupancy Map

To generate the map of the environment, the Occupancy Map Generator extension is used. This extension enables the creation of a binary map that indicates whether a specific scene area is occupied or unoccupied. The map is generated at a specific height corresponding to the Lidar's elevation above the ground.

The Occupancy Map Generator produces two essential outputs: the occupancy map's parameters, which are saved in .yaml format, and the map's corresponding .png image. The map image, as depicted in figure5.4, displays the occupied and unoccupied regions. These generated outputs are then loaded into and used by Nav2 for navigation purposes.

The .yaml file containing the occupancy map's parameters provides information about the map's resolution, origin and size. The .png image, on the other hand, visually represents the environment's occupancy information.

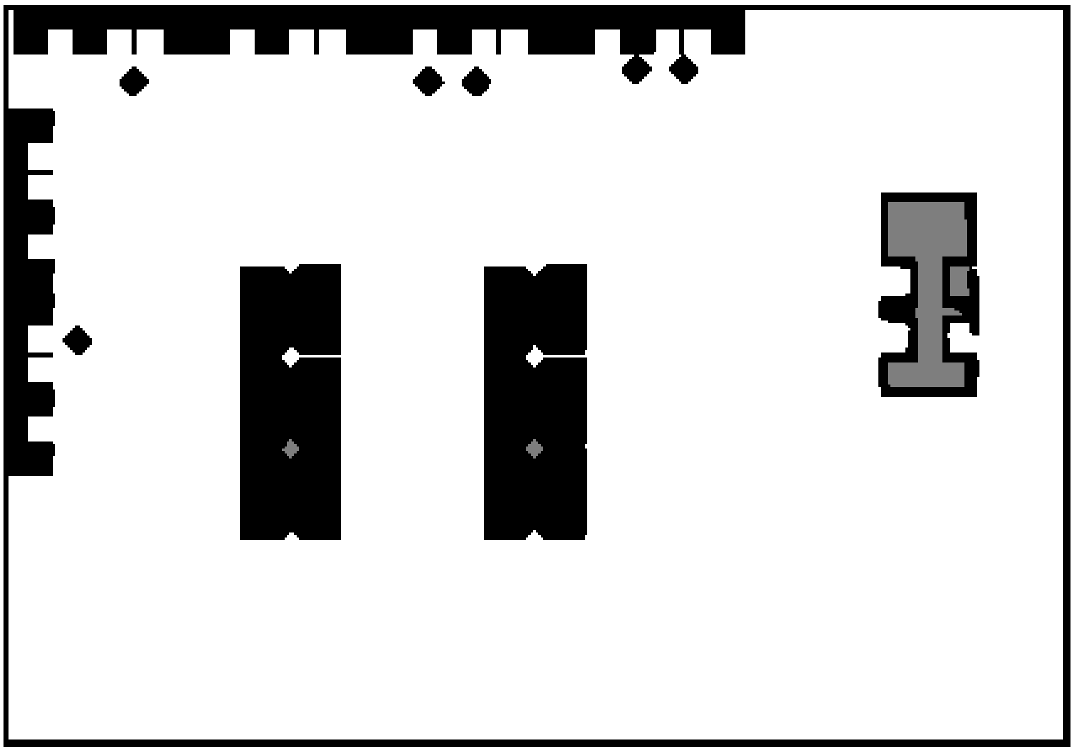


Figure 5.4: An occupancy map of the laboratory environment. The black-colored areas indicate occupied regions, which correspond to obstacles within the environment. The grey-colored areas represent unknown regions, while the white-colored areas indicate free space.

5.4 Localization and Navigation

Localization and navigation are fundamental aspects of any mobile robotic system, including mobile manipulators. Localization refers to the robot's ability to determine its position and orientation within its environment, while navigation refers to the robot's ability to plan and execute movements between locations while avoiding obstacles. These capabilities are essential for a mobile manipulator's effective and secure operation within its environment.

5.4.1 Transformations and Odometry

To enable navigation functionality, it is necessary to publish the appropriate robot transformations on the `tf` topic. Additionally, the odometry must be computed and published on the `odom` topic. Odometry is the estimation of a robot's position and orientation based on its movement and proprioceptive sensor data, as opposed to the localization system that uses the external references to localize the robot in the known environment.

Isaac Compute Odometry takes the chassis's information as input and generates its linear and angular velocities, as well as the position and orientation with respect to the World coordinate system. These values are then forwarded and published by the Publish Odometry node onto the `odom` topic.

To publish the transform between the `odom` frame and the `base_link` frame, the Publish Raw Transform Tree node is utilized. Moreover, the Publish Transform Tree node is employed to publish the static transformation between the `base_link` frame and the chassis prim frame. This transform tree encompasses the entire robot's transform hierarchy, with the `base_link` frame as the parent.

Furthermore, an additional Publish Transform Tree node is used to publish the static transformation between the `base_link` frame and the Lidar frame. All of these transformations are published on the `tf` topic, which is subscribed to by the navigation stack.

Lastly, the Publish Laser Scan node is responsible for publishing the 2D LaserScan data obtained from the Read Lidar Beams node. This node reads all the properties of the Lidar sensor, including the simulated data generated by it, and then forwards this information to the Publish Laser Scan node, which publishes the LaserScan data on `scan` topic.

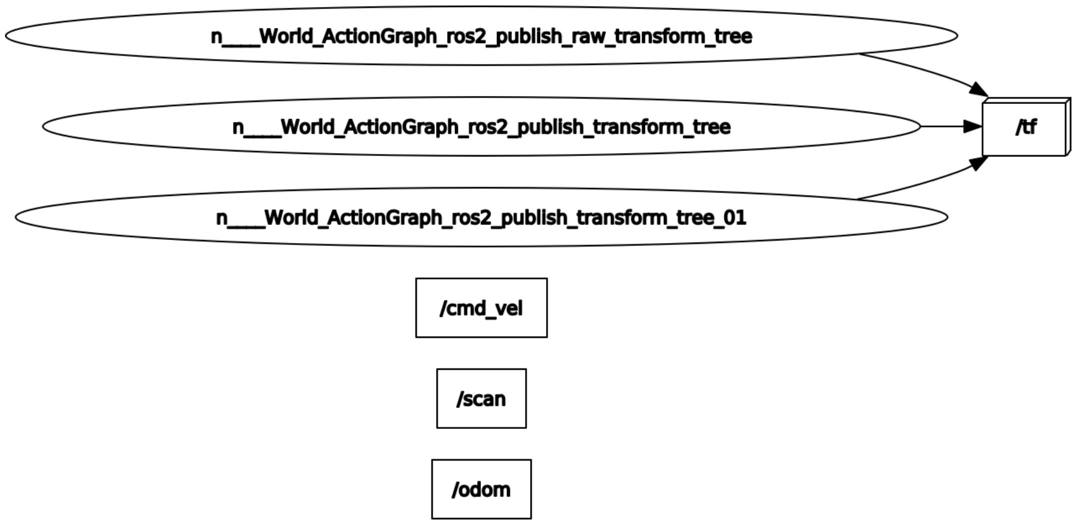

n___World_ActionGraph_ros2_publish_raw_transform_tree
n___World_ActionGraph_ros2_publish_transform_tree
n___World_ActionGraph_ros2_publish_transform_tree_01
/tf
/cmd_vel
/scan
/odom

Figure 5.5: The rqt graph provides a visual representation of the active ROS nodes and topics within the system. It offers a graphical depiction of the current state of ROS communication, specifically in this case where Nav2 is not running. As a result, it only displays the active nodes and topics within the simulated environment.

5.4.2 Localization - AMCL

AMCL (Adaptive Monte Carlo Localization) is an efficient probabilistic localization system used to enable accurate robot localization within the environment. Leveraging Lidar data published on the `scan` topic, AMCL estimates the robot's current position. Its sequential data processing approach enhances efficiency and makes it well-suited for localization tasks, eliminating the need for data storage. [20]

The impact and effectiveness of AMCL can be observed in figure 5.6, where the visualization showcases the localization process. Initially, a higher uncertainty is represented by a larger number of green particles, indicating a broader range of potential robot positions. However, as time progresses, the localization uncertainty diminishes, reflected by a reduction in the number of green particles. This visualization effectively demonstrates how AMCL progressively improves localization accuracy over time.

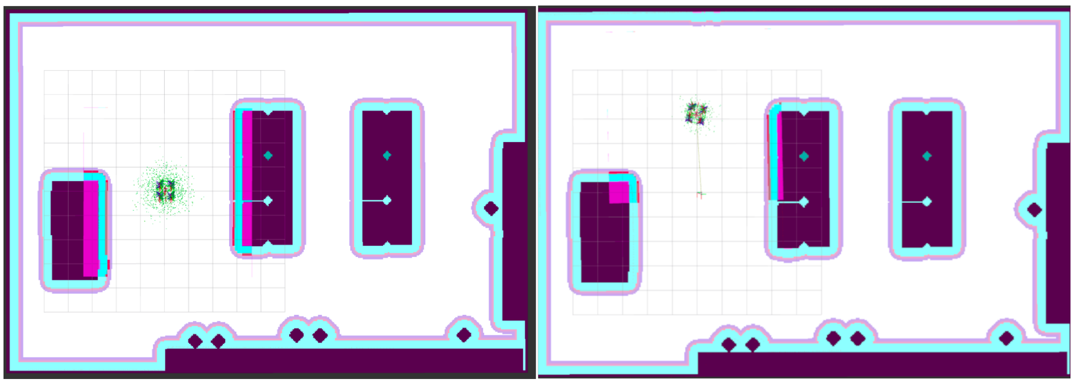


Figure 5.6: Visualization of AMCL's effect is demonstrated by showcasing the particle swarm at the beginning of navigation on the left side, and after some time on the right side. The particle swarm is represented by green dots, symbolizing the uncertainty associated with the localization process. At the initial stage of navigation, the uncertainty in the robot's localization is high, as indicated by the dense distribution of green particles. However, as time progresses, the uncertainty decreases significantly, as depicted by the reduced number of green particles. This reduction in the number of particles reflects the improved accuracy and confidence in the robot's estimated position.

5.4.3 Planner Server

The Planner Server is used to compute a feasible path from the initial pose to the goal pose. This project employs the NavFnPlanner, a robust plugin that is based on Dijkstra's algorithm, a graph-search based path planning technique. By utilizing a pre-constructed connectivity graph, it conducts a graph search to identify the shortest path, as described in the section. 2.1.2.

5.4.4 Controller Server

The Controller's main task is to determine the command velocity for a mobile robot, enabling it to navigate towards a specified goal while avoiding obstacles. It achieves this by utilizing information from both the planner server and the local costmap. This project utilizes the DWB Controller[3]. The DWB Controller generates feasible velocities and selects the ones with the highest scores, considering various metrics known as critics.

While the DWB Controller is suitable for both differential and omnidirectional robots, certain modifications need to be made in the parameters file to adapt it for use with an omnidirectional mobile platform. Specifically:

- To enable strafing movements in the y-axis, the `max_vel_y` parameter must be set to a positive value. By default, in the differential base configuration, this value is set to 0.
- Additionally, to enable acceleration in the y-direction, the `acc_lim_y` parameter should also be set to a positive value, accompanying the velocity in the y-direction.

5.5 Arm Control

To assemble the mobile manipulator, the Panda robot arm is mounted on top of the SummitXL mobile platform. The arm model is already included as one of the assets within the Isaac Sim.

To control the robot arm and generate its motion, the Moveit2 platform is utilized. The connection between the simulated robot arm and the Moveit2 platform, responsible for executing the calculated trajectories, is established using the `ros2_control` framework. PickNik Robotics has developed a ROS2 package called `topic_based_ros2_control`[4] that enables control of the robot system using ROS2 topics. For that propose, an additional Action Graph is created.

[3]https://github.com/locusrobotics/robot_navigation/tree/master/dwb_local_planner

[4]https://github.com/PickNikRobotics/topic_based_ros2_control

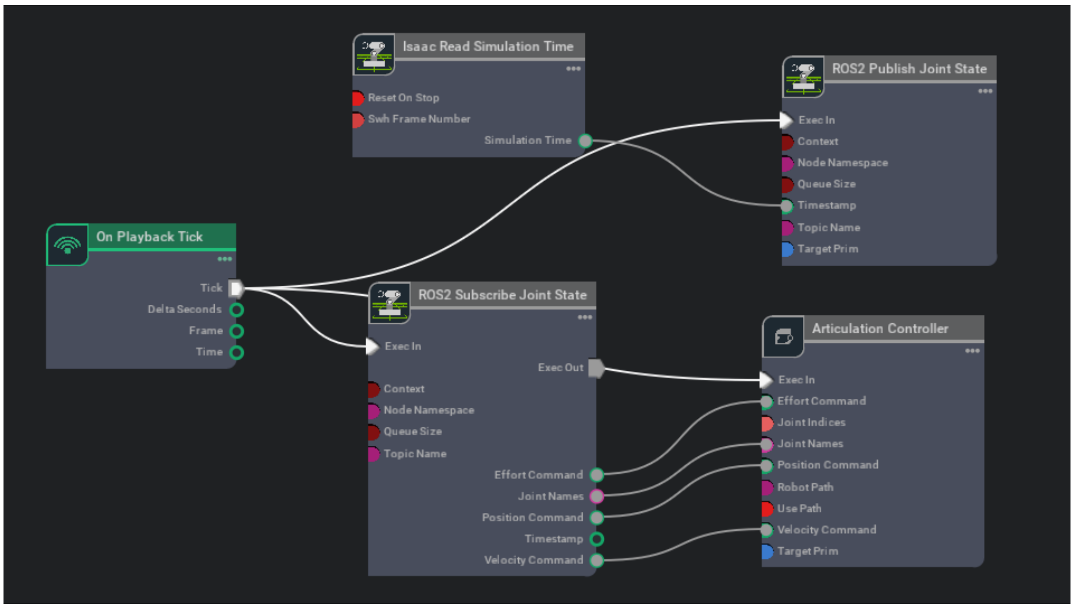


Figure 5.7: Franka Action Graph

The Action Graph includes a node that publishes the joint states of the simulated arm on the `isaac_joint_states` topic. Additionally, a subscriber is used to receive commands from Moveit via the `isaac_joint_commands` topic. These received joint values are then forwarded to the articulation controller, which actuates the joints in the simulation. The configuration of this Action Graph can be observed in figure 5.7.

To run the MoveIt2 instance, the Docker container based on the ROS2 Humble image with Ubuntu 20.04 is employed. Within this container, the `topic_based_ros2_control` ROS2 package is cloned, which is mentioned earlier. Additionally, as of the time this thesis was written, there are no official Python bindings available for Moveit2. However, a solution has been found by forking and modifying the pymoveit2[5] GitHub repository.

Pymoveit2 serves as a basic Python interface for MoveIt 2, designed to work with its ROS 2 actions and services. By leveraging these functionalities, the modified version of pymoveit2 ensures compatibility with the Isaac Sim version of the Panda robot and the available ROS 2 actions to control the robot arm and the gripper. This enables the control of the Panda robot using Python scripts, where desired joint positions, goal poses in Cartesian or joint space, as well as gripper commands, can be sent.

This approach replaces the need to rely solely on the graphical user interface (GUI) version of Moveit using the Rviz2 tool, thereby enabling the potential for development of autonomous solutions in the proposed mobile manipulator system.

[5]https://github.com/AndrejOrsula/pymoveit2

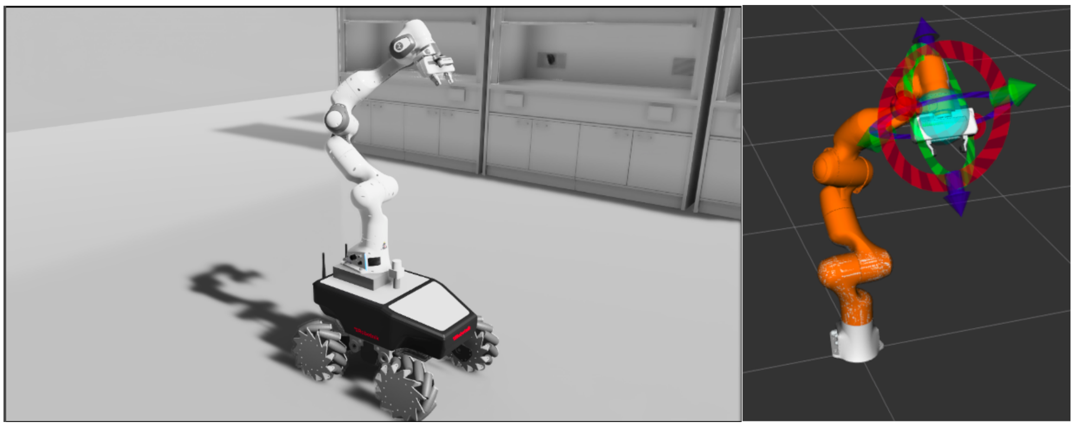


Figure 5.8: A mobile platform with a panda arm mounted in Isaac Sim on the left, controlled by the Moveit platform via the Rviz2 graphical interface on the right

5.6 System Overview

In the previous sections of this chapter, the implementation details of each component of the mobile manipulator system were discussed. This section will present an overview of the system as a whole, as shown in figure 5.9. The focus will be on the communication between three major components of the system: the Isaac Sim simulation environment, the Navigation stack, and Moveit2.

To enable the autonomous behavior of the system, both the navigation goals and arm control goals are sent using separate Python scripts, providing a more modular and flexible control approach compared to defining the goals through the rviz interface.

The defined task in this thesis can be divided into smaller sub-tasks, with the navigation of the mobile platform and manipulation of the robot arm as sub-tasks. This approach, known as separate planning as described in section 2.1.2, allows for independent low-level logic execution by different components.

The navigation goals are defined and sent via a Python script, specifying the initial pose of the mobile platform and subsequent goals. These goals are sent one by one to the Navigation stack, specifically the Planner server. The Planner server employs the Dijkstra's algorithm to compute the shortest path to goal.

At system launch, the navigation stack loads the previously generated occupancy map of the environment. The map server provides this map to the global costmap, local costmap, and AMCL. The global costmap represents the environment where the system operates, assigning cost values to each tile based on characteristics derived from the occupancy map. On the other hand, the local costmap incorporates sensor data to provide information about the robot's immediate surroundings. This allows for real-time obstacle detection and optimization of the global plan.

AMCL uses the odometry data sent from the simulation and the environment map to localize the robot during the navigation process, ensuring accurate position estimation.

The controller server computes command velocities, which are published on the `cmd_vel` topic. These velocities are then used by the holonomic controller in the simulated environment to solve the inverse kinematics of the robot. The resulting joint velocities are actuated using the articulation controller.

In the event of navigation issues or obstacles, the recovery server is called upon. It applies predefined recovery behaviors to help the robot navigate out of such situations.

Upon successful achievement of the initial goal, the Arm control script is executed. This script contains predefined joint positions to place the arm in the grasping position. Subsequently, the close gripper command is issued, followed by specifying the desired joint positions to pick up the object.

The motion planning for the robot arm is done by the MoveIt2 platform. When the script involves robot arm movement, it defines the move group name for the robot arm joints and the goal joint positions. MoveIt2's move group interface calculate trajectories using RRT Connect algorithm. The resulting joint commands for arm movements are published on the `isaac_joint_command` topic. These commands are then used in the Isaac sim to actuate the arm joints using the articulation controller, enabling the execution of planned arm trajectories.

The motion planing of the robot arm is carried out using the Moveit2 platform. If the script calls for the robot arm movement, it has defined the move group name for the robot arm joints and the goal joint positions. in this case the Panda arm controller action is called which is based on the Simple controller manager taht calles the move group interfaces which calculates the trajectories based on the RRT connect algorithm, and publishes the joint commands on the `isaac_joint_command` topic. Which are then used in the issac sim to actuate the joints using the articulation controller.

In the case of gripper movement, the Panda hand controller action is called. It utilizes the panda hand move group, specifically designed for controlling the gripper. The move group interface in MoveIt2 calculates trajectories for the gripper movement. The resulting commands for gripper movements are also published on the `isaac_joint_command` topic. These commands are then used in the Isaac sim to actuate the gripper using the articulation controller, enabling the robot to perform the desired gripper movements in coordination with the arm motion.

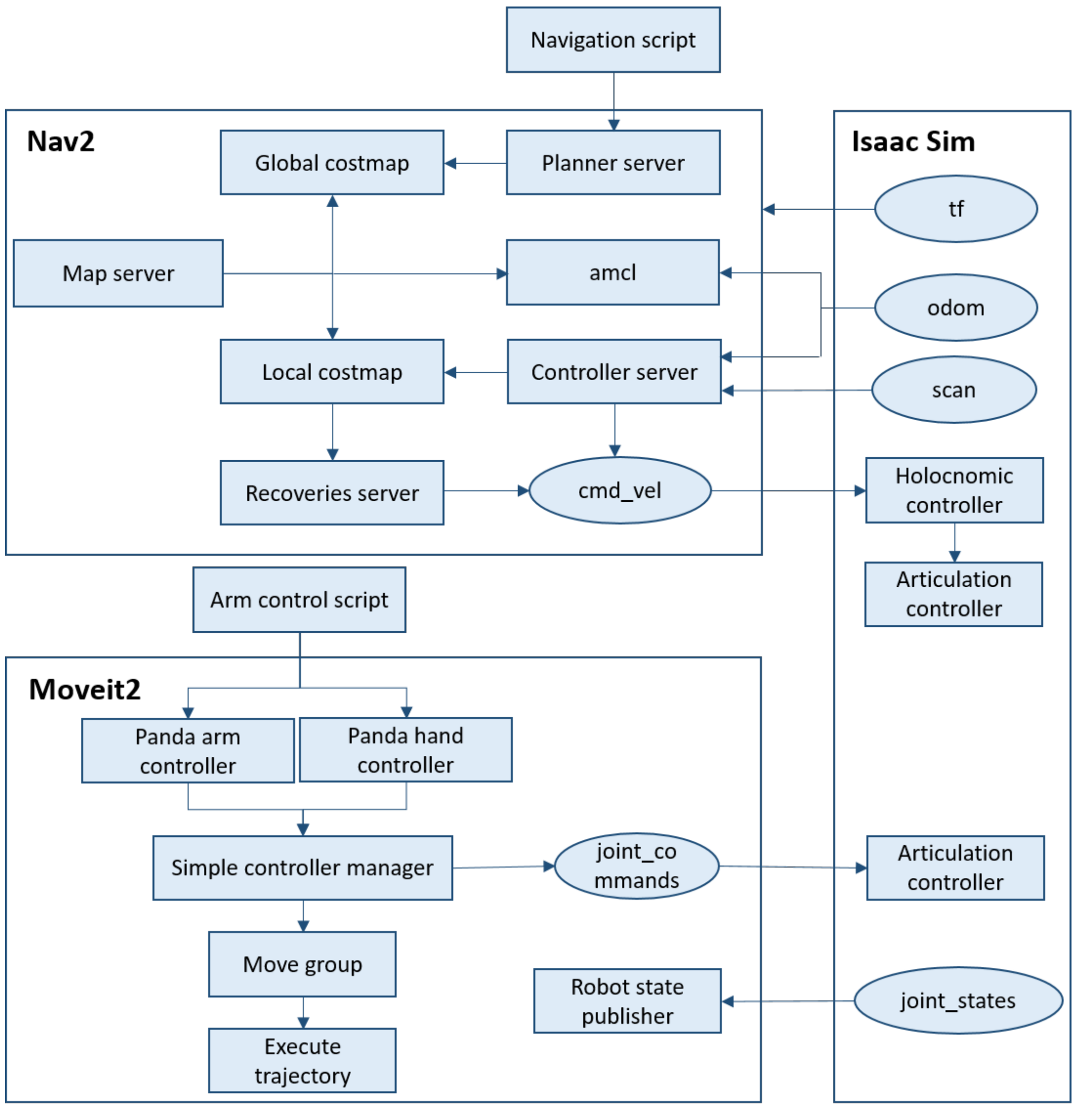


Figure 5.9: Implementation of the proposed system based on the ROS framework. The diagram showcases the three main components of the system: the Navigation stack, Isaac Sim, and Moveit2. The figure highlights the flow of information between these systems and the execution of actions set by the Python scripts. ROS topics are represented by ellipses.

Chapter 6

Testing

In this chapter, the proposed mobile manipulator system will undergo testing. The testing process will include separate evaluations of the navigation system and the robot arm manipulation system.

All tests will be conducted on a machine with Ubuntu 20.04 with the ROS2 Foxy version installed. The simulation tool used will be Isaac Sim version 2022.2.1. Additionally, Moveit2 will be executed in a Docker container on Ubuntu 20.04 with the ROS2 Humble version installed.

The videos of all experiments can be found in the appendix B, along with additional videos documenting the system's development. The code developed for this project is available in the GitHub repository in the appendix A.

6.1 Navigation System

The initial experiments will focus on evaluating the performance of the navigation system in two specific scenarios. The first scenario involves navigation in open, obstacle-free space, while the second scenario assesses the system's obstacle avoidance functionality. In the obstacle avoidance scenario, a novel obstacle will be introduced into the environment that is not included in the provided occupancy map.

6.1.1 Navigating Through Frespace

The objective of this experiment is to evaluate the performance of the navigation system in a obstacle-free scenario and evaluate the capabilities of the used DWB controllers on the omnidirectional mobile platform.

The task for this experiment is straightforward: to navigate from the initial position to the goal position in a section of the environment without any obstacles.

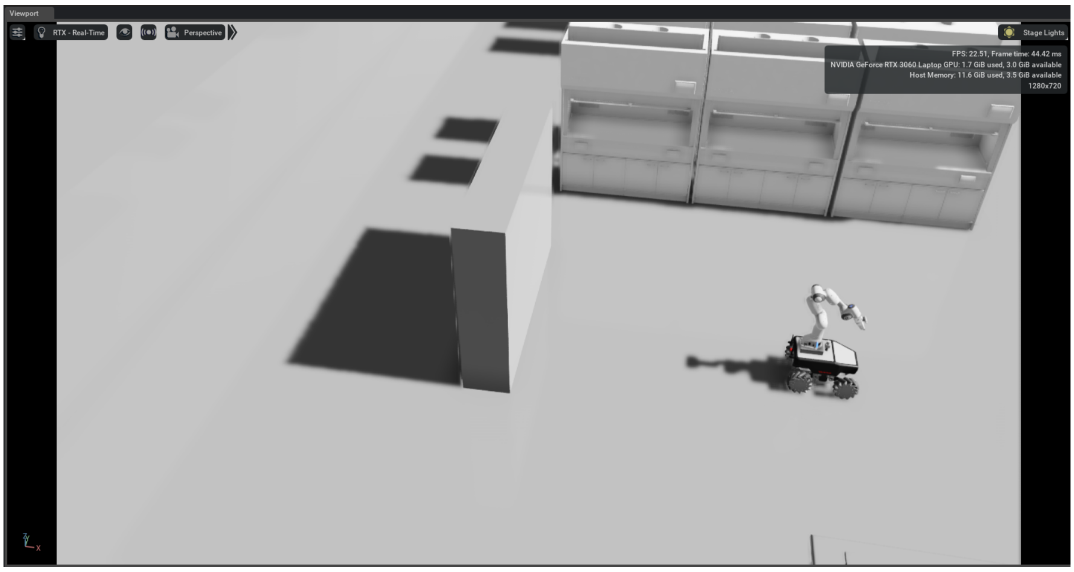
The results from the first experiment, confirmed the anticipated behavior of the DWB controller. The mobile manipulator effectively reached its destination by employing omnidirectional motion whenever it resulted in the shortest path.

6.1.2 Obstacle Avoidance

This experiments aims to assess the performance of the navigation system in detecting and effectively avoiding novel obstacles. It consists of two distinct experiments. The first experiment involves an obstacle that partially obstructs the path, requiring the navigation system to navigate around it. The second experiment presents a scenario where the obstacle completely blocks the planned path, requiring the mobile platform to find an alternative route to reach the desired goal position.

Experiment 1: Partial Constraint Scenario

For this experiment, an additional wall is introduced into the environment, which is not included in the generated occupancy map. Figure 6.1 illustrates the setup of the first experiment, where the wall is positioned near the initial location of the robot. The objective is to navigate to the goal position located just behind the wall.



```
FPS: 22.51, Frame time: 44.42 ms
NVIDIA GeForce RTX 3060 Laptop GPU: 1.7 GiB used, 3.0 GiB available
Host Memory: 11.6 GiB used, 3.5 GiB available
1280x720
```

Figure 6.1: Setup of the scene for the first experiment. The position of the wall is shown as well as the initial position of the robot.

The summary of this experiment is shown in figure 6.2. The figure demonstrates that the navigation system performed as expected. The obstacle was successfully detected, and a new path was continuously generated to guide the robot around the obstacle. Consequently, the robot effectively navigated to the goal position while avoiding the novel obstacle in the environment.

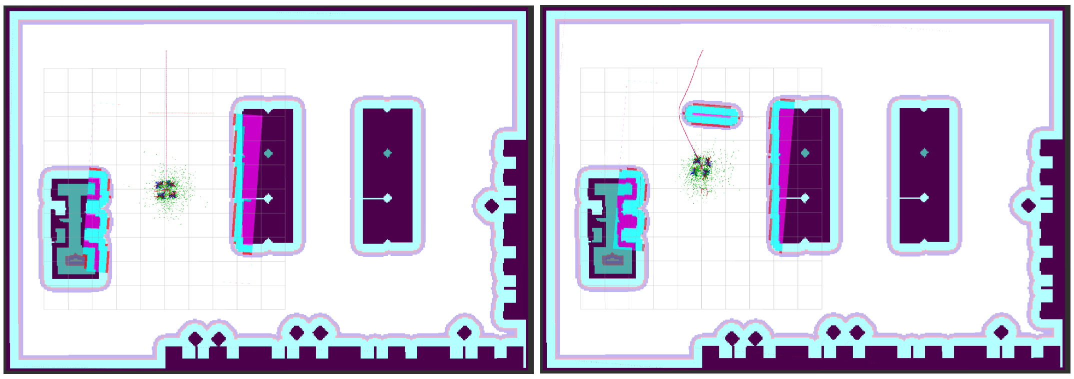
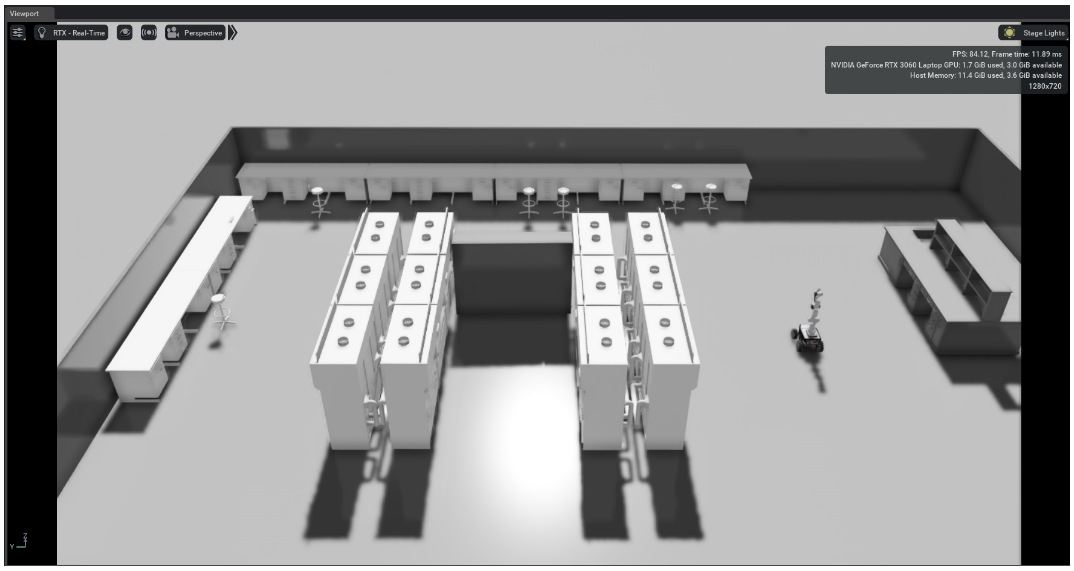


Figure 6.2: Outcome of the first experiment displayed in rviz2. On the left, the initial position of the robot is depicted, along with the initially generated path towards the goal position. On the right, the moment when the local planner detects an obstacle in the environment and generates new path that avoids the newly encountered obstacle.

Experiment 2: Complete Constraint Scenario

In this experiment, a novel wall is strategically positioned to remain undetectable from the robot's initial location. This wall is placed between two sets of fume tanks, effectively blocking the robot's passage through that specific aisle. The setup of this experiment is visually presented in Figure 6.3.



```
FPS: 84.12, Frame time: 11.89 ms
NVIDIA GeForce RTX 3060 Laptop GPU: 1.7 GiB used, 3.0 GiB available
Host Memory: 11.4 GiB used, 3.6 GiB available
1280x720
```

Figure 6.3: Setup of the scene for the second experiment. The wall is placed between two sets of fume tanks, blocking the passage through that aisle.

The goal position is deliberately chosen so that the shortest planned path by the planner server intersects with this wall. Once the wall is detected through the LIDAR sensor, a new path should be generated to guide the robot along an alternate route, bypassing the obstruction, and ultimately reaching the goal position via an unobstructed pathway.

The outcome of this experiment is depicted in Figure 6.4. Initially, the robot successfully navigated from its starting position towards the wall within the environment. The wall was detected by the LIDAR sensor, and the local costmap was appropriately updated. At this point, a new path was generated as expected, guiding the robot to take a different route to reach the desired destination.

However, as the robot moved away from the wall, it gradually moved out of the reach of the local costmap. Consequently, the new path generation process treated the wall as if it was disappearing from the environment. This led to a loop in the robot's behavior, where it would initially take a correct route, but then return to its previous position once the obstacle seemingly vanished from the local costmap.

Due to this issue, the navigation system failed to successfully guide the robot to the goal position in this scenario.

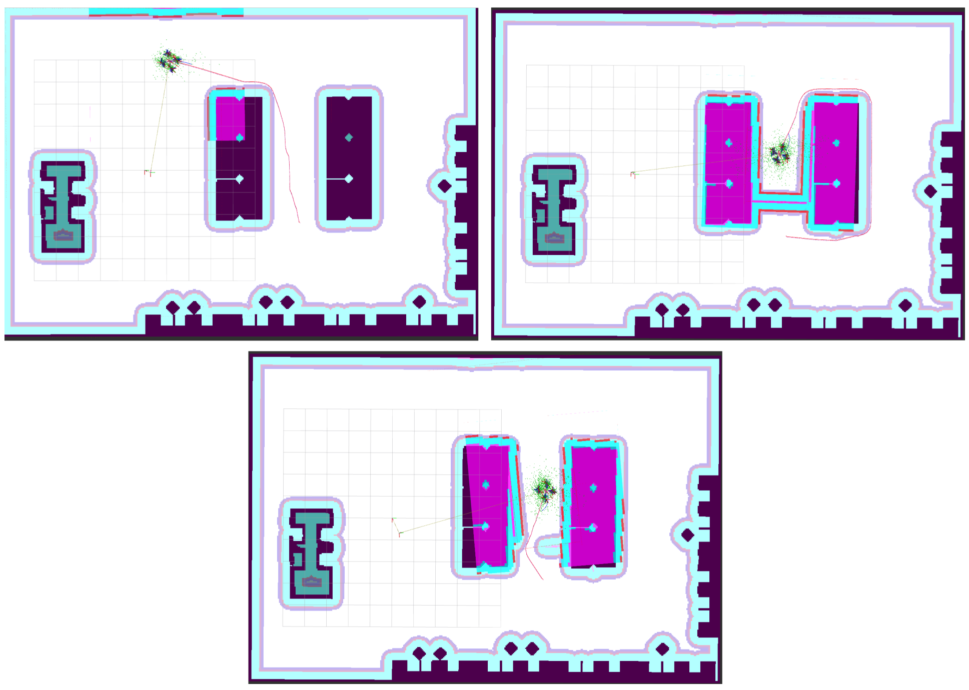


Figure 6.4: Outcome of the second experiment shown in rviz2. In the top-left, the initial position of the robot and the initial path are shown. In the top-right, the moment when the local costmap is updated, detecting an obstacle, and generating a new feasible path that avoids the obstacle. In the bottom section, the moment when the local costmap clears a portion of the obstacle. However, this leads to the generation of a new path that becomes infeasible, causing the robot to get stuck in a loop.

6.1.3 Evaluation of the Navigation System

To summarize the results of the navigation system testing, it can be stated that the system generally functions well. However, in specific scenarios as described earlier, it exhibits failures. One notable observation during testing was the controller's tendency to approach very close to the corners of obstacles in an attempt to achieve the shortest possible path. This behavior often led to the robot getting stuck, as it perceived a collision with the obstacle. This issue can be mitigated by exploring different values for the critical parameters of the DWB controller, particularly `PathAlign.scale` and `PathDist.scale`, which determine the prioritization of the robot's orientation with respect to the global path and the prioritization of staying close to the global path, respectively.

Moreover, increasing the inflation radius of both the global and local costmaps can further minimize the chances of collision when navigating corners of objects. However, it is important to consider potential undesired consequences, as a larger inflation radius may prevent the robot from getting close enough to the table where the grasping object is located, in order for the robot arm to grasp it.

Lastly, the issue observed in experiment 2, where the obstacle is removed from the local costmap as the robot moves away from it, could potentially be addressed by setting the `clearing` parameter of the obstacle layer to false. This would prevent the obstacle from being removed from the costmap, even if it is not detected by the laser scans. However, it is worth noting that this solution may work effectively for static obstacle scenarios but may not be suitable for dynamic obstacle settings. It is one potential solution among various available options to address this issue by adjusting different parameters of the navigation stack. Therefore, further investigation and future works of this project may explore additional approaches to resolve this issue.

6.2 Robot Arm System

This section includes tests to evaluate the functionalities of the motion planning implementation using Moveit2 and executing trajectories in the simulation. The experiments conducted will assess different aspects of the system, such as testing predefined joint position movement using Python scripts, evaluating motion execution in front of the table, and verifying object grasping functionality.

6.2.1 Moving to Predefined Joint Positions

This experiment aims to assess the communication between Moveit2 and the simulated robot, as well as evaluate the performance of motion planning using Moveit2. The objective is straightforward: using a Python script based on the pymoveit2 library, three predefined goal joint positions will be set. The script will then initiate a ROS action to request motion planning computation via Moveit2, followed by the execution of the computed trajectories.

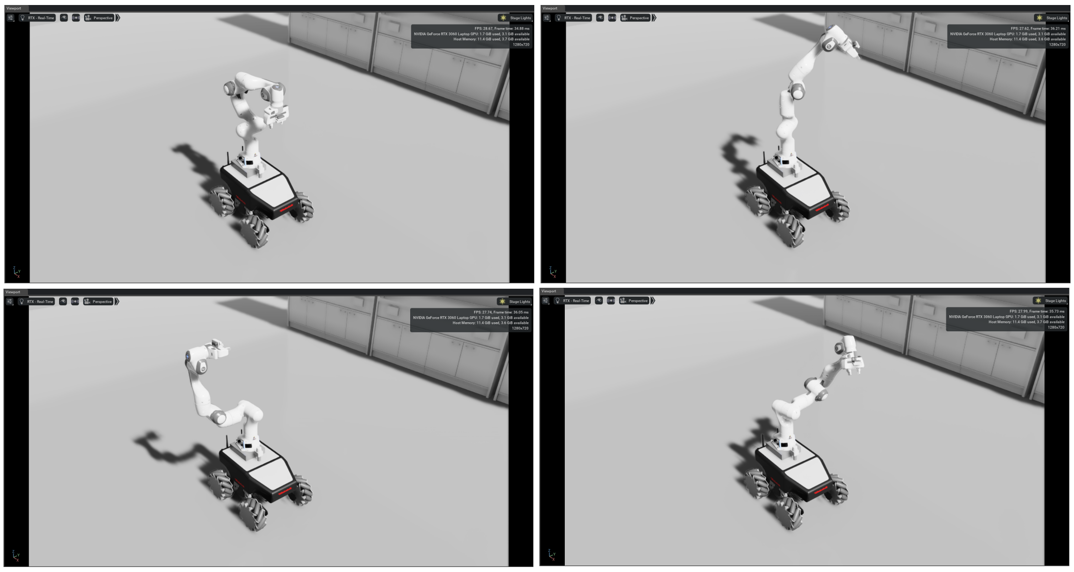


Figure 6.5: Outcome of the experiment shown in Isaac Sim. The top-left image shows the starting psoe of the robot. The subsequent images show the completion of each joint configuration

The experiment's outcome, depicted in Figure 6.5, demonstrates the successful generation and execution of trajectories in the simulation. Based on this observation, it can be concluded that the motion planning system effectively generated the desired trajectories and executed them accurately within the simulated environment.

6.2.2 Motion Planning and Execution with Collision Objects

This experiment aims to assess the system's motion planning capability in front of the table where the grasping object is located. The goal of this test is to evaluate the system's awareness of collision objects in the environment and its ability to plan and execute collision-free motions.

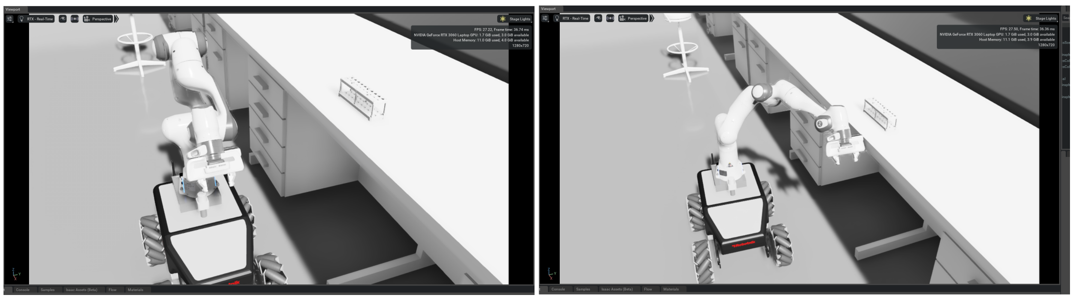


Figure 6.6: Outcome of the experiment shown in Isaac Sim. The left image presents the initial pose of the robot, while the right image displays the collision of the robot arm with the table.

In this test scenario, the mobile manipulator will be positioned next to the table. A specific joint configuration will be executed, intentionally designed to result in a collision with the table.

The results of this experiment, shown in figure 6.6, confirm the expected limitations of the motion planning interface. It was observed that the interface lacked awareness of colliding objects within the environment, as demonstrated in this test. Moreover, it can be concluded that similar outcomes would occur if the robot arm were to collide with the mobile platform, highlighting the absence of perception within the robot arm system.

6.2.3 Grasping

This experiment aims to assess the grasping capabilities of the system, specifically evaluating its ability to grasp and lift the flask rack, which corresponds to the overall task of this project.

The experimental setup mirrors the previous one, with the robot positioned next to a table where the flask rack is placed. Using predefined joint values, the system plans and executes the approach motion towards the grasping object. Subsequently, the close gripper action is called, followed by the execution of the lifting motion.

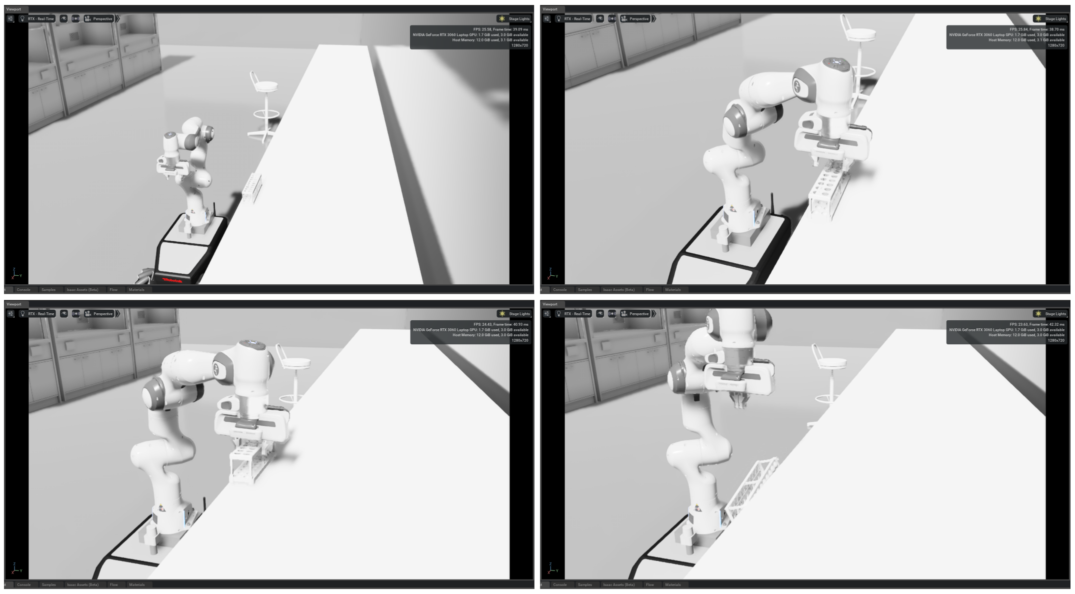


Figure 6.7: Outcome of the experiment shown in Isaac Sim. In the top-left, the starting pose of the robot. The top-right shows the approach of the robot arm towards the grasping object. In the bottom-left, an unsuccessful grasp attempt. Finally, in the bottom-right, the slipage of the grasping object during the picking motion.

The outcome of this test, as depicted in figure 6.7, reveals that the motion planner successfully generates all the necessary arm control motions, ensuring the correct approach to the grasping object. However, an error arises when the gripper command is issued to close. The motion planning interface encounters difficulty as the gripper's fingers fail to reach their intended goal due to the presence of the grasping object between them. Consequently, the grasping quality is compromised, resulting in the object slipping from the gripper during the lifting motion.

6.2.4 Evaluation of the Robot Arm System

The testing conducted on the robot arm system demonstrates that its overall basic functionality is operational. The communication between Moveit2 and Isaac Sim is successfully established, allowing the executed trajectories to be transferred to the simulated robot via ROS2 topics. However, the second test revealed a significant limitation in the system's perception capabilities, leading to its failure to detect the table and avoid collisions. Addressing this issue requires exploring several approaches.

The first approach involves defining the scene in the Moveit interface, including all the collision objects present in the environment. This would make the motion planner aware of the scene and enable collision avoidance. However, implementing this solution efficiently becomes challenging for a mobile manipulator system like ours.

The second approach involves calibrating the robot arm when the mobile manipulator reaches the table. This can be achieved through force-based calibration, as demonstrated in the work of Burger et al.[3] where the robot used a touch-sensitive 6-point calibration method to enhance its positioning precision. The robot touches six points on a cube that is associated with each experimental station to find the position and orientation of the cube relative to the robot.

The final approach entails implementing vision-based perception. Upon reaching the table, images of the scene can be captured, and machine learning algorithms can be employed for object recognition, object detection, and grasping pose estimation. Either of these last two approaches is necessary to achieve the fully autonomous behavior of the system. Currently, all goal joint positions are predefined, and even minor changes in the scene can lead to unexpected motion planning outcomes.

In the final test, an error was encountered during the execution of the close gripper command. This occurred because the presence of the grasping object prevented the gripper's fingers from fully closing, resulting in a poor-quality grasp. Currently, the implementation of the gripper command defines the gripper's open and close states based on the position of the fingers. To address this issue, a modification is required to shift from a position-based definition to an effort-based one. This means defining a force threshold that determines the gripper's close state. By adopting this approach, the Moveit interface would no longer return an error when the fingers are not fully closed. Moreover, by defining the force threshold, the grasping quality can be improved, as sufficient force would be applied to firmly grip the object and prevent it from slipping while being lifted.

Chapter 7

Discussion

This section discusses the potential improvements for the design of the proposed robotic system outlined in this work. Subsequently, an evaluation of the project objectives is provided, followed by suggestions for further development and improvement of this work.

7.1 System Design

The thesis focuses on the integration of the Franka Emika Panda robot arm and Summit XL omnidirectional mobile platform to create a robotic mobile manipulator specifically designed for the MAPs use case in a laboratory environment. The motion planning approach adopted for this system is separate planning, where the mobile platform and the robot arm are treated as independent systems for motion planning purposes. To facilitate this, the navigation stack 2 is implemented to handle navigation for the mobile platform, while Moveit2 is utilized for motion planning and trajectory execution for the robot arm.

Although the proposed system is designed and evaluated in a simulated environment, certain practical considerations need to be addressed for real-life implementation. In particular, the Panda robot arm requires a power supply, and its control necessitates the connection of a controller unit. These aspects were not addressed in the simulation. Therefore, in order to make the setup functional in real-world scenarios, a controller unit and power inverter need to be mounted on the mobile platform.

Given the limited space and payload capacity of 50 kg of the mobile platform, this setup may not be optimal. As a result, alternative solutions provided by robotics companies such as Clearpath Robotics and Robotnik can serve as examples. For instance, the XL-GEN mobile manipulator utilizes the same Summit XL mobile platform but mounts a lightweight Kinova arm on the front-top area, incorporating embedded controllers at each actuator. This arrangement saves space for additional equipment or transportation.

Another noteworthy example is the Ridgeback Franka mobile manipulator from Clearpath Robotics, which combines the Panda arm with the Ridgeback mobile platform. The Ridgeback platform possesses omnidirectional capabilities as the Summit XL, but it boasts a significantly higher payload capacity of 100 kg. This increased payload capacity enables more efficient utilization of the system in transportation-oriented use cases.

7.2 Evaluation of Project Objectives

This section will assess the degree to which the stated project objectives from chapter 3 have been completed.

Project objective 1

The implemented mobile manipulator system has demonstrated successful integration within the simulated environment. However, as mentioned in the previous section, it is crucial to address the key components necessary for the real-life implementation of the Panda robot arm.

Project objective 2

The robotic mobile manipulator system was successfully simulated in Isaac Sim, leveraging its ROS2 bridge. The implementation included Moveit2 platform-based motion planning for the robot arm and navigation stack-based motion planning for the mobile platform. Additionally, Python scripts were utilized to achieve successful execution of various tasks, including sending navigation goals for the mobile platform, as well as goal poses and gripper commands for the robot arm.

Project objective 3

A simulated environment resembling a laboratory setting was created. This environment contains walls, fume hoods, and different tables typically found in a chemical laboratory. Additionally, a sample rack was incorporated into the simulation as a grasping object.

Project objective 4

This objective has been partially fulfilled. Thorough testing of the low-level functionalities of both the navigation system and the robot arm system was carried out and analyzed. Moreover, specific issues with the current implementation were identified, and potential approaches to address these issues were proposed. However, the overall task defined as navigating to the goal, grasping the object, transporting it to the storage location, and placing it there, remains incomplete.

Project objective 5

The GitHub repository containing the source code, installation and running instructions can be found in Appendix A, along with the documented videos showcasing the testing and development process in Appendix B.

7.3 Future Works

This section discusses planned and incomplete work, as well as future work proposed for this project.

As detected and discussed in the chapter 6, some of the functionalities of both the navigation system, and robot arm system did not perform well. There are several possible direction this work could take in order to address those issues.

- In order to achieve autonomous operation of the system, the perception system for the robot arm needs to be implemented. Currently, the lack of this functionality requires predefined goal poses for the robot arm and the position of the grasping object. This approach poses challenges when integrating with the navigation system, as it cannot guarantee perfect alignment with the navigational goal due to inherent errors in goal position alignment.

  To address this issue, future work could explore implementing a camera sensor on the robot arm along with vision-based grasping approaches. This would allow for real-time perception of the environment, enabling more accurate alignment and object detection. Alternatively, another direction to tackle this issue would involve implementing a force-based calibration system, which could enhance the system's ability to adapt and align with the goal position by utilizing feedback from force sensors.

- To address the documented issues with the navigation system, a thorough exploration of the various parameters of the system can be conducted to understand their impact on functionality. This analysis would involve documenting the effects of different parameter configurations on the system's performance.

  Additionally, testing different plugins, particularly the controller plugin, is crucial. Currently, the implemented DWB controller tends to move very close to the corners of collision objects, which is undesirable. Exploring alternative controllers designed for omnidirectional robots, such as the TEB controller, could be beneficial. By evaluating and comparing the performance of different controller options, a more suitable controller can be identified to improve the navigation system's behavior and achieve more precise and efficient movements.

Apart from addressing the current issues of this system, future improvements can be centered around implementing low-level control logic, such as state machines. These advancements would enable the system to handle more complex tasks compared to the one defined in this thesis, thereby bringing it closer to autonomously solving and assisting in real-life laboratory environments.

In addition, the implementation of digital shadow and digital twin could serve as potential extensions to this work. Digital shadow refers to a virtual representation of a physical object or system, capturing its real-time state and behavior. On the other hand, a digital twin is a more advanced concept that not only mirrors the physical object or system but also utilizes real-time data to simulate and predict its future behavior.

By incorporating digital shadow and digital twin into this system, the existing simulation setup can be used to replicate the logic and movement of the physical hardware in real life. This integration would enhance the system's capabilities by bridging the gap between the virtual and physical domains, allowing for better monitoring, analysis, and control of the laboratory environment.

Chapter 8

Conclusion

The objective of this thesis was to acquire practical knowledge of working with the Isaac Sim simulation tool and the ROS framework for controlling and simulating robotic systems. Additionally, the goal was to explore the potential of autonomous robotic systems in MAPs and provide a foundation for implementing an omnidirectional mobile manipulator in a laboratory setting, thereby contributing to CAPeX's mission of revolutionizing new material discovery.

This thesis resulted in the development of a simulated mobile manipulator system within a laboratory environment. The proposed system consists of an omnidirectional Summit XL mobile platform with a 7 DoF panda robot arm mounted on top. Furthermore, motion planning for this system was addressed by employing a separate body planning approach. The overall task was divided into smaller subtasks, with each sub-task being solved by either the robot arm or the mobile platform. ROS-based frameworks, specifically the navigation stack 2 for autonomous navigation of the mobile base and the Moveit2 platform for motion planning and trajectory execution of the robot arm, were utilized in this work.

Moreover, comprehensive testing and exploration of the system's capabilities were conducted, providing valuable insights into the workflow and identifying potential areas for future research and improvement. Although the overall task outlined in Chapter 3 was not fully accomplished, it is anticipated that this work will prove valuable for further investigations in both mobile manipulators and MAPs. It provides a functional simulation of an omnidirectional mobile manipulator and offers a modular solution for motion planning of both the mobile platform and the robot arm.

Bibliography

[1] Meysam Basiri et al. “An autonomous mobile manipulator to build outdoor structures consisting of heterogeneous brick patterns”. In: *SN Applied Sciences* 3 (May 2021). DOI: 10.1007/s42452-021-04506-7.

[2] Kate Brush. *What is a mobile robot? definition from whatis.com*. 2019. URL: https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics.

[3] Benjamin Burger et al. “A mobile robotic chemist”. In: *Nature* 583 (July 2020), pp. 237–241. DOI: 10.1038/s41586-020-2442-2.

[4] Chuang Cheng et al. “Stability Control for end effect of mobile manipulator in uneven terrain based on active disturbance rejection control”. In: *Assembly Automation* 41.3 (2021), 369–383. DOI: 10.1108/aa-10-2020-0157.

[5] Nikolaus Correll et al. *Introduction to autonomous robots: Mechanisms, sensors, actuators, and algorithms*. The MIT Press, 2022.

[6] Faza Fahleraz. “A Comparison of BFS , Dijkstra ’ s and A * Algorithm for Grid-Based PathFinding in Mobile Robots”. In: 2018.

[7] Martha M. Flores-Leonar et al. “Materials Acceleration Platforms: On the way to autonomous experimentation”. In: *Current Opinion in Green and Sustainable Chemistry* 25 (2020), p. 100370. ISSN: 2452-2236. DOI: https://doi.org/10.1016/j.cogsc.2020.100370. URL: https://www.sciencedirect.com/science/article/pii/S2452223620300596.

[8] *Isaac Sim*. 2022. URL: https://developer.nvidia.com/isaac-sim.

[9] Weria Khaksar et al. “A Low Dispersion Probabilistic Roadmaps (LD-PRM) Algorithm for Fast and Efficient Sampling-Based Motion Planning”. In: *International Journal of Advanced Robotic Systems* 10.11 (2013), p. 397. DOI: 10.5772/56973.

[10] Steven Macenski et al. “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[11] Steven Macenski et al. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[12] Mayank Mittal et al. *Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation*. 2022. arXiv: `2103.10534 [cs.RO]`.

[13] Juan J. de Pablo et al. "New frontiers for the materials genome initiative". In: *npj Computational Materials* 5.1 (2019). DOI: `10.1038/s41524-019-0173-4`.

[14] Krishna Rajan, Christopher A Bockisch, and Kamal Choudhary. "The evolution of Materials Acceleration Platforms: toward the integration of data-driven discovery and optimization". In: *Journal of Materials Science* 56.25 (2021), 15662–15677.

[15] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. "A review of mobile robots: Concepts, methods, theoretical framework, and applications". In: *International Journal of Advanced Robotic Systems* 16.2 (2019), p. 1729881419839596. DOI: `10.1177/1729881419839596`.

[16] Thushara Sandakalum and Marcelo H. Ang. "Motion Planning for Mobile Manipulators - A Systematic Review". In: *Machines* 10.2 (2022). ISSN: 2075-1702. DOI: `10.3390/machines10020097`. URL: `https://www.mdpi.com/2075-1702/10/2/97`.

[17] Martin Sereinig, Wolfgang Werth, and Lisa-Marie Faller. "A review of the challenges in Mobile manipulation: Systems design and Robocup challenges". In: *Elektrotech. Informationstechnik* 137.6 (2020), 297–308. DOI: `10.1007/s00502-020-00823-8`.

[18] Ksenia Shabalina, Artur Sagitov, and Evgeni Magid. "Comparative Analysis of Mobile Robot Wheels Design". In: *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. 2018, pp. 175–179. DOI: `10.1109/DeSE.2018.00041`.

[19] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT Press, 2011.

[20] Alexander S. Staal et al. "Towards a Collaborative Omnidirectional Mobile Robot in a Smart Cyber-Physical Environment". In: *Procedia Manufacturing* 51 (2020). 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021), pp. 193–200. ISSN: 2351-9789. DOI: `https://doi.org/10.1016/j.promfg.2020.10.028`. URL: `https://www.sciencedirect.com/science/article/pii/S2351978920318837`.

[21] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: `10.1007/s12532-020-00179-2`. URL: `https://doi.org/10.1007/s12532-020-00179-2`.

[22] Daniel P. Tabor et al. “Accelerating the discovery of materials for clean energy in the era of Smart Automation”. In: *Nature Reviews Materials* 3.5 (2018), 5–20. DOI: 10.1038/s41578-018-0005-z.

[23] Hamid Taheri, Bing Qiao, and Nurallah Ghaeminezhad. “Kinematic Model of a Four Mecanum Wheeled Mobile Robot”. In: *International Journal of Computer Applications* 113.3 (2015), pp. 6–9.

[24] Rama Vasudevan, Ghanshyam Pilania, and Prasanna Balachandran. “Machine learning for materials design and discovery”. In: *Journal of Applied Physics* 129 (Feb. 2021), p. 070401. DOI: 10.1063/5.0043300.

[25] Manman Yang et al. “Collaborative mobile industrial manipulator: A review of system architecture and applications”. In: *2019 25th International Conference on Automation and Computing (ICAC)*. 2019, pp. 1–6. DOI: 10.23919/IConAC.2019.8895183.

[26] Chengmin Zhou, Bingding Huang, and Pasi Fränti. *A review of motion planning algorithms for intelligent robotics*. 2021. arXiv: 2102.02376 [cs.RO].

Appendix A

GitHub Repository

You can access the repository for the project by accessing the following link: `https://github.com/mld95/Summit_ws`

Appendix B

Video Documentation

The video documentation of the development process as well as the testing of the system can be viewed at the following YouTube playlist: `https://www.youtube.com/playlist?list=PLr33eX-Y3mhEYlLT0T_l3PN-Kq0lvLbCF`