AUTONOMOUS NAVIGATION OF QUADRUPED INTEGRATED WITH MANIPULATOR

BY

VENKATA CHINTHALAPATI
B.S., OSMANIA UNIVERSITY, 2019
M.S., UNIVERSITY OF ILLINOIS CHICAGO, 2024

THESIS

Submitted as partial fulfillment of the requirements
for the degree of **Master of Science in Mechanical**
in the Graduate College of the
University of Illinois at Chicago, 2024
Chicago, Illinois

DEFENSE COMMITTEE:

Pranav Bhounsule, Ph.D., Chair

Micheal Scott, Ph.D.

Young Soo Park, Ph.D.

DEDICATION

I would like to dedicate this thesis to my parents Ramachandra Murthy and Rama Devi. Thank you.

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisor, Prof.Pranav Bhounsule for his invaluable help and guidance throughout my Master's program. Your continued support helped me to accomplish tasks both academically and professionally. Thank you to all members of the RAM Lab. Your support and assistance have been crucial, and without your help, completing my work would have been much more difficult. Lastly, I want to extend heartful thanks to my friends and family, for your continued support and understanding.

AUTONOMOUS NAVIGATION OF QUADRUPED INTEGRATED WITH MANIPULATOR

Venkata Chinthalapati, M.Sc.
The University of Illinois Chicago, 2024

Supervising Professor: Pranav Bhounsule, Ph.D.

The focus of this research is the assembly of the Open Manipulator X on the Unitree Go1 quadruped robot, enhancing its capabilities to execute complex manipulation tasks in unstructured environments. By integrating various sensors, including the camera already present on the Go1, a SLAMTECH LiDAR, and the Go1's IMU sensor, a robust motion planning and control framework is developed. This framework enables the torque-controlled quadrupedal robot to perform dynamic locomotion while simultaneously executing manipulation tasks, thereby achieving a high level of autonomy without the need for human intervention. The framework is verified on the real robot by performing tasks such as reaching the specified end effector point by navigating through obstacles present in the environment. The experimental results demonstrate the effectiveness of the proposed system in navigating and performing tasks in diverse and challenging scenarios, showcasing advancements in the field of autonomous robotics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In the wake of significant scientific and technological advancements, it has become possible to engineer robotic systems that are not only more intelligent but also capable of a broader spectrum of applications. These systems are constantly being developed to mitigate human limitations and enhance efficiency, cost-effectiveness, and safety across multiple sectors, including industry, agriculture, surveillance, security, and even domestic settings. Historically, research in mobile robotics has emphasized the development of fully autonomous systems or those requiring minimal human intervention, thereby enhancing productivity and safety while potentially offering economic benefits.

Quadrupeds, or four-legged robots, are engineered to mimic the locomotion of animals such as dogs and horses, making them particularly effective in traversing complex terrains like rubble, caves, and construction sites. Equipped with advanced sensors like cameras and LIDAR, these robots can replace humans in hazardous environments, collecting crucial data and providing first responders with vital pre-entry information to enhance their safety.

Integrating a manipulative limb into a quadruped significantly broadens its application in real-world scenarios. These robots can navigate and interact with their environment, performing tasks that are dangerous, physically demanding, or require intricate manual dexterity. Such capabilities make quadruped manipulators versatile, functioning efficiently both indoors and outdoors under various conditions. They are programmable to execute tasks with high precision, either autonomously or in collaboration with other systems.

Several leading companies are exploring the use of these robots for package handling and other logistics applications. The integration of a manipulative arm with a quadruped extends its functionality, allowing it to undertake more complex tasks.

1.1 Thesis Contribution

In this thesis, the integration of the Go1 quadruped robot with the Open Manipulator X and sophisticated sensory technology—cameras and LIDAR—is explored. The primary aim of this integration is to empower the robot to autonomously navigate to a predetermined end-effector point. This capability is crucial for tasks where objects must be transported from one location to another, a common requirement in logistics, delivery services, and operations in challenging or dangerous environments.

The autonomous navigation system allows the robot to locate and move toward specific coordinates or objects within its environment. This functionality is particularly useful in scenarios such as warehouse operations, where the robot can pick up and transport goods from storage to shipping areas without human intervention. Additionally, in hazardous or inaccessible locations—such as disaster sites, contaminated areas, or extreme environments like deep mines or arctic stations—the robot's ability to operate independently reduces the risk to human life and increases operational efficiency.

Moreover, in remote or extreme climatic conditions where human presence is limited, such as polar research stations or desert outposts, the robot can perform regular supply runs, transporting food, medicine, or equipment between locations. Its robust design and

autonomous capabilities ensure that operations can continue smoothly, despite adverse weather conditions or challenging terrains.

Thus, the integration of autonomous navigation and manipulation capabilities in the Go1 quadruped robot with Open Manipulator X not only enhances its utility in conventional settings but also opens up possibilities for its application in areas where traditional robotic solutions are impractical. This makes it a valuable asset for industries and sectors requiring reliable, autonomous transport and handling of goods in complex and unstructured environments.

CHAPTER 2: BACKGROUND AND RELATED WORK

2.1 Quadruped Manipulators

Quadruped robots have garnered significant attention from researchers in recent decades due to their adaptability and agility. These are more adept at traversing obstacles, such as boulders or holes, and executing locomotion in unpredictable and unstructured environments than wheeled/tracked robots. In recent years, these robots have demonstrated significant potential for real-world applications in a variety of sectors, including education, construction surveillance, parcel delivery, search and rescue, and inspection.

2.1.1 Using Leg as Arm

One strategy institutions have explored to address manipulation challenges is the leg-arm approach. In this method, the legs of the robot are multi-functional, allowing them to perform some manipulation tasks. When necessary, the robot adjusts its position so that a leg can reach the target object. ETH Zurich applied this technique to ANYmal, a quadruped robot developed by ANYbotics, enabling it to press elevator buttons. However, the robot can only manipulate while stationary, as it cannot walk and manipulate simultaneously. The manipulation capabilities are limited to the area the leg can reach. A similar implementation by Xin et al. allowed for tasks such as pressing buttons, inspecting pipes, and sensing surfaces. Nonetheless, this change in the robot's kinematic configuration poses significant challenges in maintaining the stability of its center of mass (CoM) and controlling the overall system. Additionally, the motion

planner requires a complex optimization process to consistently provide the most effective solution.

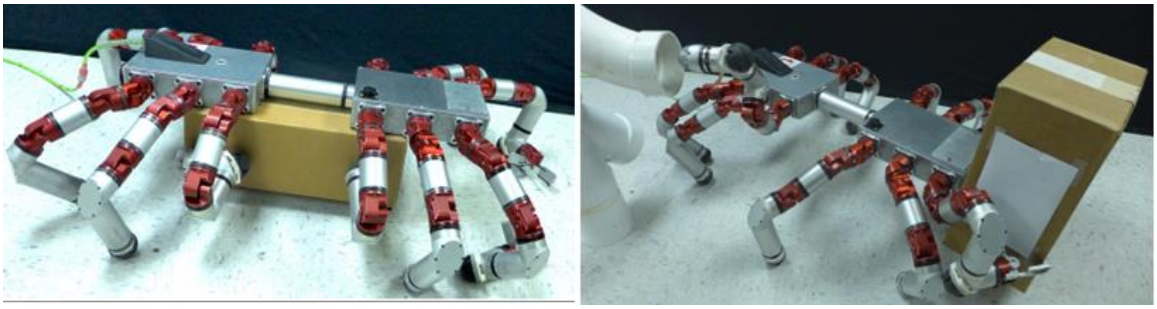

Figure 1: Dodecapod robot using its many limbs for manipulation or inspection tasks.

2.1.2 Integrate External Arm

A mobile manipulator refers to a system that combines a robotic manipulator with a mobile platform. When the mobile base is a quadruped robot, the system is known as a quadruped manipulator. While mobile platforms offer the advantage of covering large work areas, they generally lack manipulation capabilities. On the other hand, robotic arms excel in manipulation tasks but are confined to stationary workstations. By integrating a quadruped base with a manipulator arm, a more advanced system is created, offering benefits such as increased reach and maneuverability. However, this combination also introduces challenges, including the complexity of control due to the higher number of degrees of freedom (DoF) and the increased risk of instability in the overall system.

Figure 2: Spot (Left), Aliengo (Right)

2.2 Motion Planning

Mobile manipulators must carefully manage the movement of their platform and address various constraints depending on the specific goals during motion planning. One critical factor is ensuring the robot does not collide with obstacles in its environment. Additionally, the system's structural and operational boundaries, such as limitations in joint angles, torque, and movement ranges, must be considered. These constraints can vary based on the robot's design and the task it is performing.

Because these robots often have a high number of degrees of freedom (DoF), they are considered kinematically redundant. This means that there are numerous ways for the manipulator's end-effector to reach the same position, leading to a wide array of possible solutions. While this redundancy provides flexibility, it also increases the complexity of the control algorithms, making motion planning far more challenging. The system has to choose the optimal solution out of many, which often involves a balance between efficiency, stability, and task-specific requirements. This complexity is a major factor in why motion planning remains a significant hurdle for achieving effective loco-manipulation in mobile robots.

2.2.1 Independent Systems

In this method, the mobile platform and the manipulator are treated as independent systems, with motion planning done sequentially—first for one component and then for the other. This allows the use of existing algorithms that are typically applied to individual systems. When executing manipulation tasks, the quadruped remains stationary.

However, there are several key observations regarding the Separate System approach:

- If one subsystem is poorly positioned, it may prevent the manipulator from reaching its target.
- Achieving the best possible result for each subsystem individually does not necessarily lead to an optimal outcome for the entire system.
- Despite these limitations, this method tends to produce more stable and reliable results because of the simpler, modular planning process.

By approaching the platform and manipulator separately, the complexity is reduced, but it may limit overall system efficiency and performance in certain scenarios.

2.2.1 Integrated Systems

Another approach is to treat the integration of the platform and manipulator as a unified, complex system. This method enables simultaneous execution of various tasks, such as manipulator control, torso adjustment, managing joint constraints, and navigating environmental challenges. However, it significantly increases the complexity of the system due to the need to handle the interactions between the mobile base and the arm, as well as the unpredictable disturbances caused by the manipulator's movement.

This integrated approach fully leverages the system's capabilities, allowing for sophisticated, coordinated actions. However, it presents several substantial challenges:

- It demands considerably more computational power and energy to manage the intricacies of the combined system.
- The generated motion paths are often not optimal, requiring additional optimization algorithms to refine them.
- If the integration is not carefully executed, the solutions can be unstable and more vulnerable to external disturbances.
- Achieving a robust and stable solution is more difficult, as improper implementation can lead to instability throughout the robot.

While this approach offers the potential for advanced functionality and efficiency, it also introduces significant technical hurdles that must be carefully managed to avoid compromising the system's stability and performance.

2.3 ROS

ROS is an open-source, meta-operating system for building robotic applications. It provides the typical services of an operating system (OS), including hardware abstraction, low-level device control, message-passing between processes, and package management. It also provides tools and libraries for writing, building, and running code across multiple computers. Since it is not an actual OS, it has to be installed on an existing one, such as Ubuntu, one of the Linux distributions. Following is a list of basic ROS terminology and architectural components:

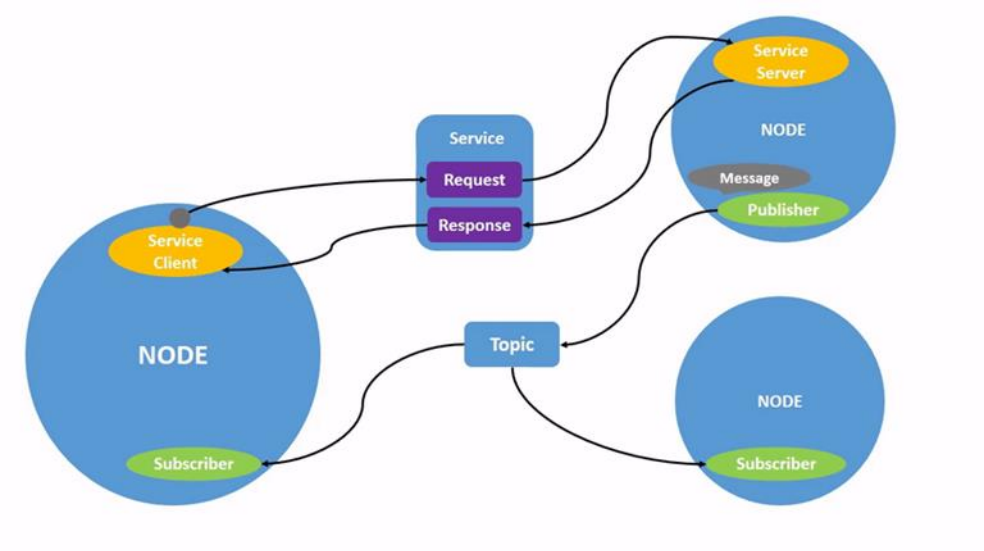

Figure 3: ROS Communication Architecture. Messages are published and subscribed between nodes and topics for communication. Service requests and response communication between server and client.

- Nodes: The base unit in ROS is called a node. Nodes are in charge of handling devices or computing algorithms - each node for a separate task. Nodes can communicate with each other using topics or services. ROS software is distributed in packages. A single package is usually developed for performing one type of task and can contain one or multiple nodes.
- Message: A message is a data structure used for communication between nodes. A simple specification language defines messages and can include various data types such as integers, floats, and arrays. For example, a message might contain sensor readings, commands, or state information.
- Master: The ROS Master provides naming and registration services to the nodes in a ROS-based system. It acts as a central manager that keeps track of all active nodes, their publications, and their subscriptions. The Master is essential for nodes to discover each other and communicate.

- Topics: In ROS, topic is a data stream used to exchange information between nodes. Topics are used to send frequent messages of one type, such as sensor readouts or information on motor goal speed. Each topic is registered under a unique name and with a defined message type. Nodes can connect to the topic to either publish messages or subscribe to them. For a given topic, one node can not publish and subscribe to it at the same time, but there are no restrictions on the number of different nodes publishing or subscribing.
- Services: A service is a synchronous communication mechanism in ROS. It allows one node to send a request to another node and receive a response. Services are defined by a pair of messages: one for the request and one for the response. This is useful for tasks that require a request-reply interaction.

ROS acts as a meta-operating system for robots as it provides hardware abstraction, low-level device control, inter-processes message-passing, and package management. The main advantage of ROS is that it allows manipulation of sensor data of the robot as a labeled abstract data stream, called topic, without having to deal with hardware drivers. This makes the programming of robots much easier for software developers as they do not have to deal with hardware drivers and interfaces. Also, ROS provides many high-level applications such as arm controllers, face tracking, mapping, localization, and path planning.

Mobile robot navigation generally requires solutions for three different problems: mapping, localization, and path planning. In ROS, the Navigation Stack plays a role in integrating all the functions necessary for autonomous navigation.

2.3.1 SLAM

Simultaneous Localization And Mapping (SLAM) is an important algorithm that allows the robot to acknowledge the obstacles around it and localize itself. When combined with some other methods, such as path planning, it is possible to allow robots to navigate unknown or partially known environments. ROS has a package that performs SLAM and path planning along with other functionalities for navigation like Navigation Stack: Gmapping and hector_mapping.

Both Gmapping and hector_mapping are implementations of SLAM, a technique that consists of mapping an environment at the same time that the robot is moving, in other words, while the robot navigates through an environment, it gathers information from the environment through its sensors and generates a map. This way you have a mobile base able not only to generate a map of an unknown environment but also to update the existent map, thus enabling the use of the device in more generic environments, not immune to changes.

The difference between Gmapping and hector_mapping is that the first one takes in account the odometry information to generate and update the map and the robot's pose. However, the robot needs to have proprioceptive sensors, which makes the usage of it hard for some robots (e.g. flying robots). The odometry information is interesting because they are able to aid on the generation of more precise maps, since understanding the robot kinematics we can estimate its pose.

Kinematics is influenced, basically, by the way that the devices that guarantee the robot's movement are assembled. Some examples of mechanic features that influence the

kinematics are: the wheel type, the number of wheels, the wheel's positioning and the angle at which they are disposed.

2.3.2 ROS Navigation Stack

To achieve the navigation task, the Navigation Stack is used to integrate the mapping, localization, and path planning together. It takes in information from odometry, sensor streams, and the goal position to produce safe velocity commands and send it to the mobile base (Fig.4). The odometry comes through nav_msgs/Odometry message over ROS which stores an estimate of the position and velocity of a robot in free space to determine the robot's location. The sensor information comes through either sensor_msgs/LaserScan or sensor_msgs/PointCloud messages over ROS to avoid any obstacles. The goal is sent to the navigation stack by geometry_msgs/PoseStamped message. The navigation stack sends the velocity commands through geometry_msgs/ Twist message on /cmd_vel topic.

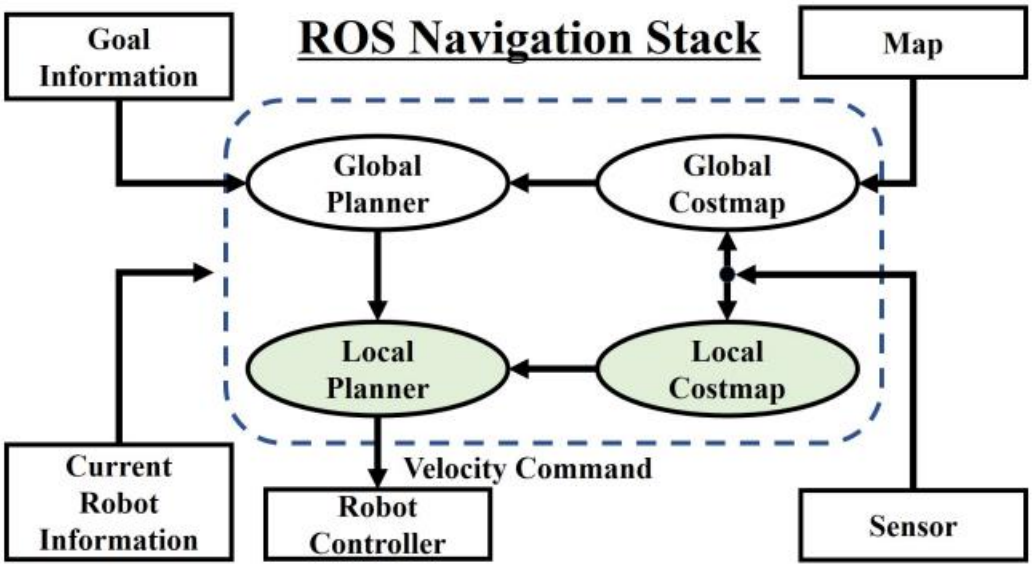

Figure 4: ROS Navigation Stack

2.3.3 Mapping

ROS provides a wrapper for OpenSlam's Gmapping. A particle filter-based mapping approach is used by the gmapping package to build an occupancy grid map. Gmapping algorithm will estimate the Lidar pose firstly based on the previous map and motion model. Then it will compute the weight according to the sensor observation to resample and update the map of particle. It will execute these steps in a cycle to complete mapping.

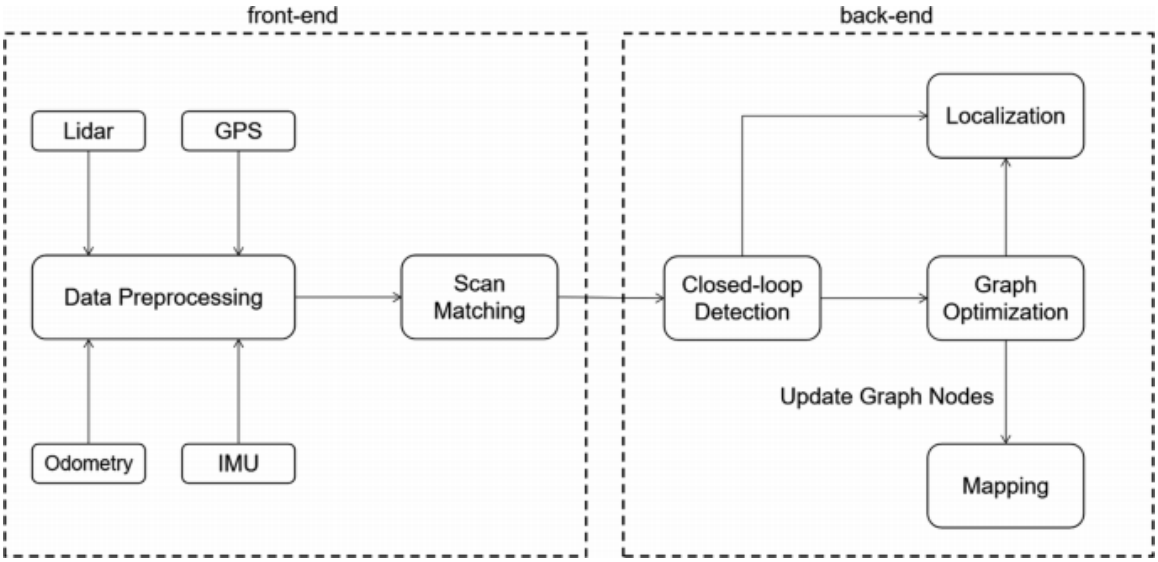

Figure 5: Overview of Front-End and Back-End involved in SLAM

Then a package named map_server could be used to save that map. The maps are stored in a pair of files: YAML file and image file. The YAML file describes the map meta-data and names the image file. The image file encodes the occupancy data.

When initialized without a prior map, the robot only knows about the obstacles detected by its sensors and can avoid these seen obstacles. For unknown areas, the robot will generate an optimistic global path that may encounter unseen obstacles. In such cases, the robot can re-plan its path to navigate around the newly detected obstacles.

2.3.4 AMCL Localization

The localization part is solved in the amcl package using an Adaptive Monte Carlo Localization which is also based on particle filters, it is a probabilistic localization system of a robot's movement in 2D space, and it adopts the particle filter to track the position and orientation of a robot in the known map. To let the robot move to an accurate position, you can adopt Adaptive Monte Carlo Localization to adjust the robot's position, which works like putting the particles uniformly on the map, and then these particles will gather in an area after algorithm calculation. The more the particles in an area, the greater the probability that the robot is in this area.

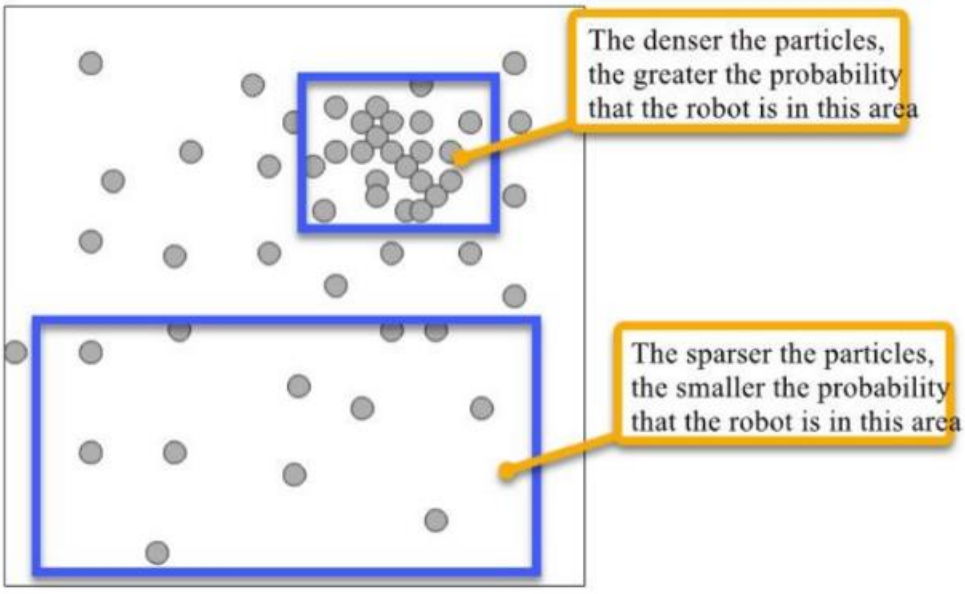

Figure 6: AMCL Localization – Probability Distribution of Robot's position based on Particle density

2.3.5 Gmapping

Gmapping, as well as amcl, is a localization system, but unlike amcl, it runs on an unknown environment, performing Simultaneous Localization and Mapping (SLAM). It

creates a 2D occupancy grid map using the robot pose and the laser data (or converted data, i.e. Kinect data). It works over the Odom to map transformation, therefore it does not need the map nor IMU information, needing only the odometry.

2.3.6 Sensors and Controller

These blocks of the system overview are in respect to the hardware-software interaction and, as indicated, are platform specific nodes. The odometry source and the base controller blocks are specific to the robot you are using, since the first one is usually published using the wheel encoders data and the second one is responsible for taking the velocity data from the cmd_vel topic and assuring that the robot reproduces these velocities.

2.3.7 Local and Global Costmaps

The local and global 2D costmaps are the topics containing the information that represents the projection of the obstacles in a 2D plane (the floor), as well as a security inflation radius, an area around the obstacles that guarantees that the robot will not collide with any objects, no matter what is its orientation. These projections are associated to a cost, and the robot's objective is to achieve the navigation goal by creating a path with the least possible cost. While the global costmap represents the whole environment, the local costmap is, in general, a scrolling window that moves in the global costmap about the robot's current position.

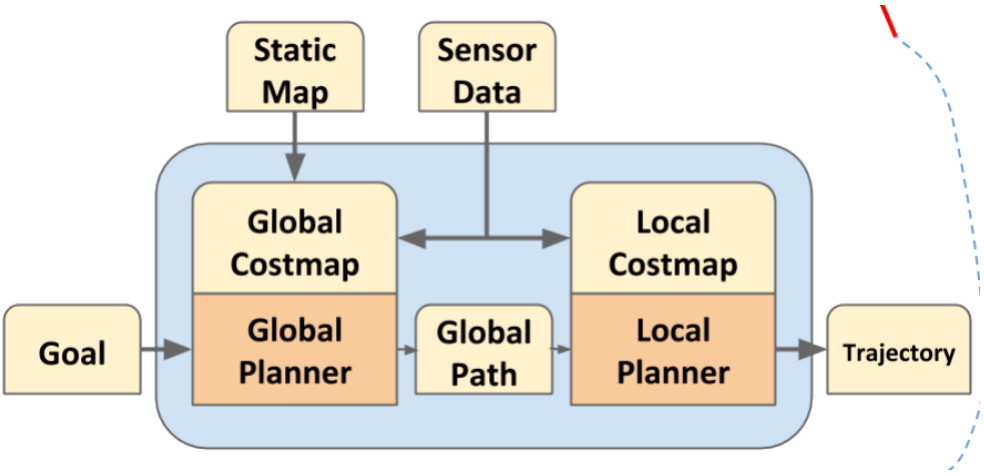

Figure 7: Integration of Local and Global Costmaps with Path Planners for trajectory planning

2.4 Global Planner

The global path planner in ROS operates on the global_costmap, which is generally initialized from a prior static map, then it could be updated frequently based on the value of the update_ frequency parameter. The global path planner generates a long-term plan from the start or current position to the goal position before the robot starts moving. It will be seeded with the costmap and the start and goal positions. These start and goal positions are expressed by their x and y coordinates. A grid-based global planner that can use Dijkstra's algorithm or A* algorithm to compute the shortest collision-free path for a robot is obtained in the global_planner package.

2.5 Local Planner

The local path planner or the controller in ROS operates on the local_costmap, which only uses local sensor information to build an obstacle map and dynamically updated with sensor data. It takes the generated plan from the global planner, and it will try to

follow it as close as possible considering the kinematics and dynamics of the robot as well as any moving obstacles information in the local_costmap. ROS provides implementation of two local path planning algorithms namely the Trajectory Rollout and the Dynamic Window Approach (DWA) in the package base_local_planner.

2.5.1 DWA Algorithm

DWA (Dynamic-Window Approach) is an algorithm that mainly samples multiple sets of velocity in velocity space (v, w), and predicts their trajectories in a specific time according to the robot dynamics model. Then these trajectories will be scored by the evaluation function, and the optimal trajectory is picked to propel the robot to move. DWA algorithm converts the position control of the robot into speed control. Before it predicts the robot's motion trajectory under speed mode, the robot's motion model should be analyzed first. v(t) and w(t) refer to the linear and angular velocities in the world coordinate system. In the sampling period Δt, if the robot's displacement is small and it executes uniform linear motion, the robot motion model is

$$x(t) = x(t-1) + v(t) \cdot \Delta t \cdot \cos\big(\theta(t-1)\big)$$

$$y(t) = y(t-1) + v(t) \cdot \Delta t \cdot sin\big(\theta(t-1)\big)$$

$$\theta(t) = \theta(t-1) + w(t) \cdot \Delta t$$

x(t), y(t), and θ(t) are the pose of the robot under the world coordinate system at t moment.

2.5.2 TEB Algorithm

TEB (Timed Elastic Band) algorithm optimizes the motion trajectory by correcting the initial trajectory of the global path planning. It mainly optimizes the distance between the

robot and the obstacle, the length of the path, and the execution time of the trajectory. TEB algorithm describes the path planning problem as a multi-objective optimization problem, which means that it will optimize the minimized execution time of the trajectory, and the constraint that keeps a certain distance between the robot and the obstacle, and follows the motion dynamics. As most of the optimization targets are local and only related to consecutive states of the robot, this optimization is targeted at sparse models.

The improved framework of the robot control system is as follows. N discrete poses with time information constitute the trajectory generated by the TEB algorithm. And then G2O (General Graph Optimization) algorithm is used to optimize these poses so that the generated trajectory is the shortest, the least time-consuming, and the farthest from obstacles. The speed and acceleration are limited to make the trajectory meet the requirements of robot kinematics.

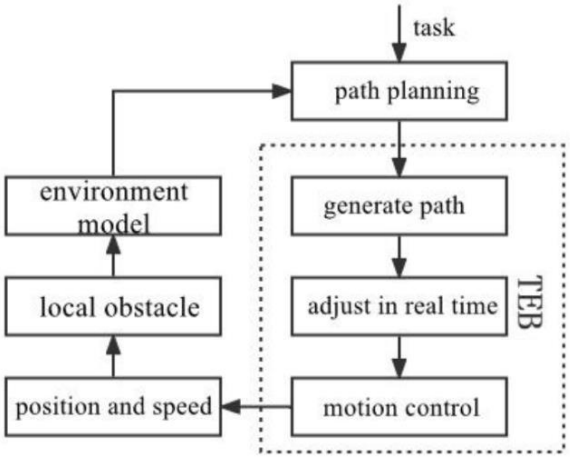

Figure 8: TEB Local Planner General Graph Optimization model framework

The coordinate of the center of the robot as well as the turning direction determine its pose in the environment. Firstly, define the robot pose

$$Xa = (x_i, y_i, \beta_i)^T \epsilon R^2 \times S^1$$

$$\tau = \{\Delta T_i\}_{i=0,1,\dots,n-1}$$

$$B := (Q, \tau)$$

The TEB algorithm sets the posture and time interval as the variables to be optimized and solves the optimal path under dynamic constraints. These dynamic constraints include velocity and acceleration limits, path lengths, distances between obstacles and the robot, and how long the robot will run on a trajectory. Then, the optimal path Q is obtained by setting the weighted multi-objective function.

Considering the characteristics and performance of both the Dynamic-Window Approach (DWA) and the Timed Elastic Band (TEB) algorithm, the TEB algorithm is selected for our application due to its superior adaptability and optimization capabilities in dynamic environments. While DWA is effective in static settings by sampling velocities and selecting optimal trajectories based on a robot's dynamic model, it can struggle with moving obstacles as it focuses primarily on short-term predictions without extensive future trajectory considerations. In contrast, the TEB algorithm excels in scenarios with dynamic obstacles by continuously optimizing the robot's trajectory through multi-objective optimization. It accounts for various dynamic constraints such as velocity, acceleration, path length, and obstacle distance, ensuring the trajectory is both safe and efficient. By using the General Graph Optimization (G2O) algorithm, TEB ensures that the generated path is the shortest, least time-consuming, and maintains a safe distance from obstacles. These capabilities make TEB a more robust and reliable choice for

environments where obstacles are in motion, providing enhanced performance and safety for the robot's navigation tasks.

2.6 Unitree Go1

The Unitree Go1 is a cutting-edge quadruped robot designed by Unitree Robotics, characterized by its highly dynamic and adaptable locomotion. Each of its four torque-controlled legs features three degrees of freedom, enabling the robot to walk, trot, and run with remarkable stability across varied terrains. This advanced locomotion capability is ideal for navigating complex environments.

The Go1 is equipped with an array of sensors that enhance its perception abilities, including five fisheye stereo depth cameras and IMUs (Inertial Measurement Units), which furnish the robot with comprehensive environmental awareness. These sensors empower the Go1 to adeptly handle tasks such as obstacle avoidance, terrain mapping, and autonomous navigation. Furthermore, the robot is capable of carrying a payload of up to 5 kilograms, making it suitable for a variety of applications. It also incorporates substantial computing power with 3x Nvidia Jetson Nanos and one Raspberry Pi, which are on the same network. They process the extensive sensory data and support complex computational tasks required for its operations.

Figure 9: Unitree Go1 Quadruped Robot

Items	Unit	Unitree Go1
Pay Load	kg	5
Max Speed	m/sec	3.7
DOF		12
Power Output	V	24
Depth Camera		5
Ultrasonic Sensors		4
Controllers		4 (3 Nano + 1 Raspberry Pi)

Table 1: Unitree Go1 Specifications

The version bought was the developer's version which comes with an onboard personal computer (PC) and allows connections to multiple interfaces, for example, High Definition Multimedia Interface (HDMI), Ethernet, and Universal Serial Bus (USB). Regarding software, the onboard PC allows remote connections and new implementations can be made. The underlying interface supports C/C++ and ROS.

2.7 Open Manipulator-X

The Open Manipulator-X is a versatile, open-source robotic arm developed by ROBOTIS powered by Dynamixel Servo Motors. It features 4 degrees of freedom (DoF) in its standard configuration. These include three rotational joints and one end-effector joint,

enabling the arm to perform various movements. The design allows for easy manipulation and precise control of objects in three-dimensional space.

The Open Manipulator-X is fully compatible with ROS (Robot Operating System), providing access to a wide range of libraries and tools for robotic control, simulation, and programming. The arm can be controlled using MoveIt, a ROS-based software for motion planning, manipulation, and control.

Items	Unit	Open Manipulator-X
Actuator		Dynamixel XM430-W350-T
Input Voltage	V	12
DOF		5 (4 DOF + 1DOF Gripper)
Payload	g	500
Reach	mm	380

Table 2: Open Manipulator-X Specifications

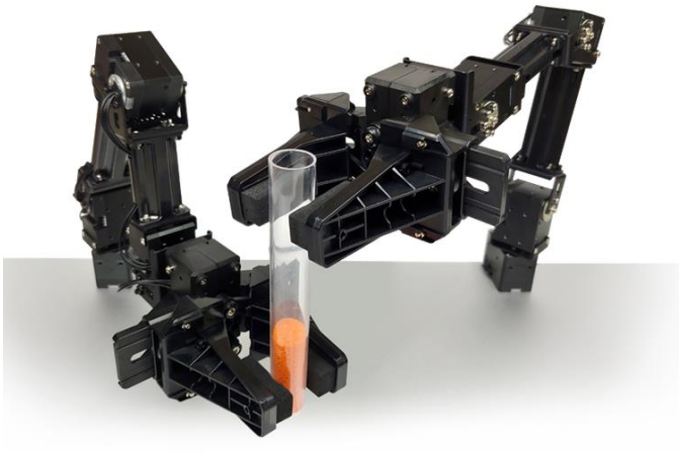

Figure 10: Robotis Open Manipulator-X

2.8 SLAMTEC Mapper M2

SLAMTEC Mapper is a new type of laser sensor introduced by SLAMTEC, which is different from traditional LIDAR. It is built with a unique SLAM optimization algorithm and high-performance LIDAR to fuse map data more than 10 times per second and construct up to 100,000 square meters of mapping area. The LIDAR carries out 9200

measurements per second, and the longest-ranging distance can reach 40 meters. The built-in processing system can process data in real-time and output high-precision maps and pose

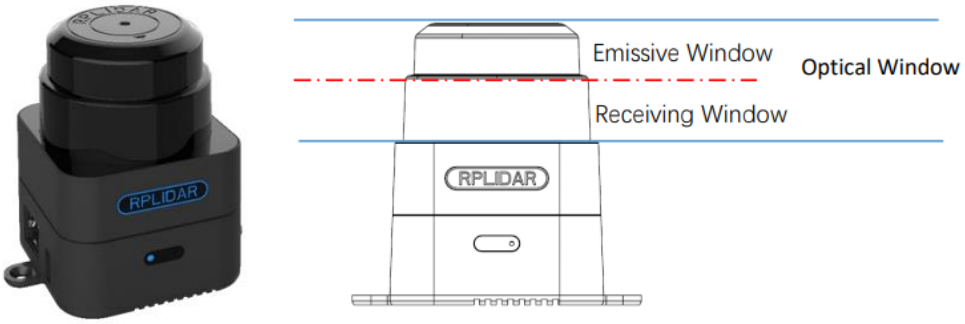

Figure 11: SLAMTEC LiDAR Mapper M2

Items	Unit	SLAMTEC Mapper M2
Distance Range	m	40
Sampling Rate	Hz	9200
Maximum Area	m	300*300
Mapping resolution	m	0.05
Power	V	5

Table 3: SLAMTEC Mapper M2 Specifications

CHAPTER 3: SETUP AND METHODOLOGY

Initially, The Jetson Nano computer was first equipped with the Linux Ubuntu 18.04 operating system. However, because of performance problems experienced during the first configuration, the environment was later built up on a virtual machine that is hosted on a personal computer to guarantee stability and ease of management. Subsequently, ROS Melodic was installed using Debian packages, adhering to the specifications outlined in the official ROS manual. A dedicated workspace, named catkin_ws, was established to promote organized source code administration and efficiently separate development from other system components.

The workspace includes several critical packages essential for the operation and integration of the Go1 quadruped robot with the Open Manipulator X, as outlined below:

- **Unitree Go1 Packages**: Sourced from Unitree Robotics, this collection of packages provides comprehensive support for operating the Go1 robot. It includes drivers and interfaces for cameras, ultrasonic sensors, and other navigation-related sensors. These packages are designed for both simulated environments and real-world applications, including a tailored interface for the Gazebo simulator which significantly aids in development and testing.
- **OpenManipulator-X Packages**: These packages, provided by ROBOTIS, contain all necessary URDF files and manipulator control software required to integrate and operate the Open Manipulator X. This integration is crucial for achieving precise manipulation capabilities.

- **Slamtec M2 Lidar Mapper Packages**: Obtained directly from Slamtec's official website, these packages are integral for navigation. They include all necessary configuration files for the move base framework and the TEB (Timed Elastic Band) path planner, which is utilized for dynamic and efficient path planning based on the robot’s and obstacles' velocities.

The files in these packages are further modified to the current hardware and simulation setup. All the modified files are uploaded to the GitHub repository and can be found following the link: https://github.com/vchint6/pathplanning_go1

3.1 Simulation Setup

After downloading the necessary packages, the initial step involves setting up the combined system of the Unitree Go1 quadruped, LIDAR, and manipulator using the URDF files provided in the packages and including all necessary plugins for the sensors. The LIDAR and manipulator are linked to the base link of the quadruped.

A simulation environment was developed using Gazebo and integrated with ROS to test the system's robustness, assess its stability, and evaluate the motion planning algorithm based on inverse kinematics (IK). This simulation enables the visualization of the quadruped's behavior in both Gazebo and RViz.

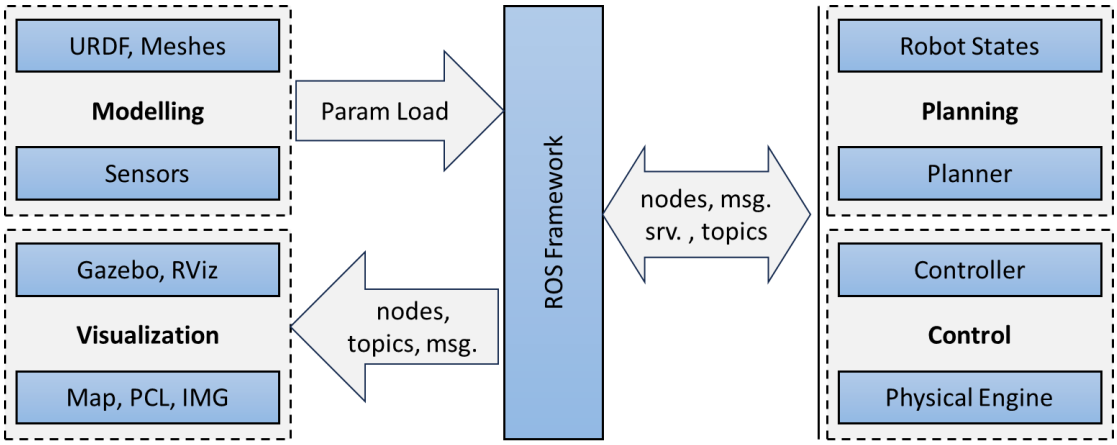

Figure 12: ROS Simulation Setup architecture

- The simulation is initiated by running a ROS launch file which provides the robot description including URDF, Meshes, Sensors, and nodes for running the controller and motion planning
- Control commands are published from the terminal or a control node to specific topics that the controllers for the quadruped and manipulator are subscribed to.
- The controller nodes publish the joint states to a topic, such as `/joint_states`. Gazebo subscribes to this topic to update the physical simulation, while RViz subscribes to visualize the joint movements.
- Gazebo updates the simulation environment based on the new joint states and simulates the physical interactions. It provides feedback by publishing sensor data and the new state of the robot on various topics.
- RViz visualizes the current state of the robot based on the joint states and other sensor data. This visualization helps in monitoring the robot's behavior and ensuring the commands are executed correctly.

3.1.1 Gazebo Setup

The purpose of integrating the manipulator on the quadruped robot is to enable it to perform pick-up and drop-off tasks, similar to delivery applications such as delivering groceries or packages. To demonstrate this capability, an environment was created in Gazebo (Fig.14) where the robot picks up a tennis ball from one point and moves it to a target point. This setup showcases the robot's ability to handle objects autonomously, utilizing all available sensors for feedback.

Similar to the real Unitree Go1 quadruped total of five depth cameras are strategically mounted on the robot. These include cameras on the face, chin, and both the left and right sides of the quadruped's trunk. These cameras provide comprehensive visual coverage of the surroundings, allowing the robot to detect objects.

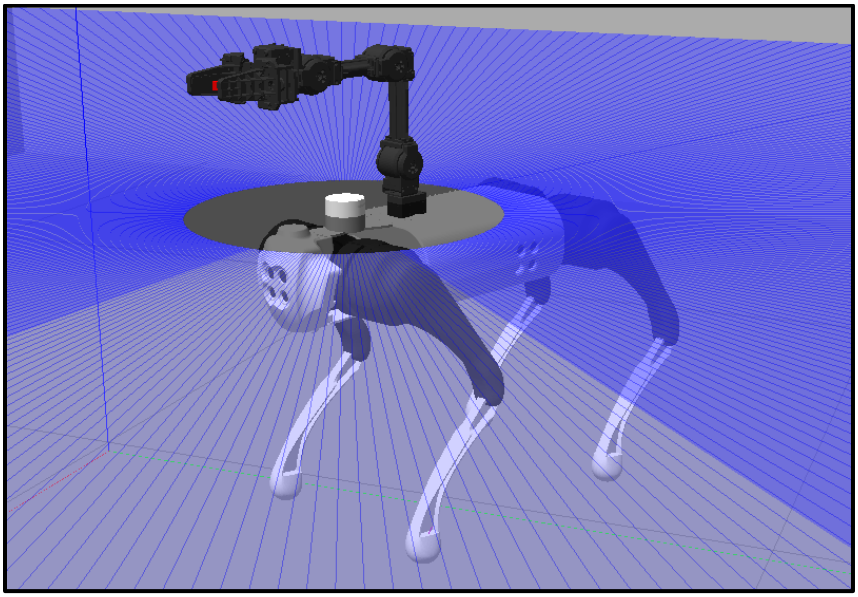

Figure 13: Gazebo URDF Model of Go1 assembled with LiDAR and Open Manipulator

A LIDAR sensor is mounted on the top of the quadruped. The LIDAR emits laser beams that can sense objects within a 20-meter range(Fig.13) . The detected objects are

represented by blue light rays in the simulation, providing a clear indication of the robot's sensory range.

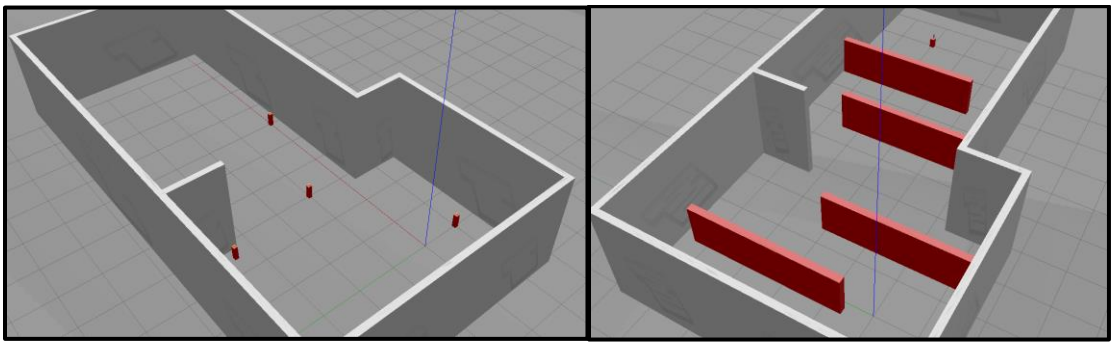

Figure 14: Gazebo Environment 1(Left), Environment 2(Right)

3.1.2 Quadruped - Motion Planning

The motion planning process for the quadruped robot involves several crucial steps, starting with the creation of a map of the environment and ending with the robot reaching the target location.

- The list of Terminal commands to run the slam package for generating the map for navigation is shown in Fig.15 and Fig.16 lists the commands to launch the Manipulator controller for manipulation to end effector position.

1. launch the simulation world -

```
roslaunch unitree_navigation my_gazebo.launch
```

2. run the go1 controller -

```
./devel/lib/unitree_guide/junior_ctrl
```

3. launch the SLAM for mapping -

```
roslaunch unitree_navigation my_slam.launch
```

4. save map -

```
rosrun map_server map_saver -f ~/catkin_ws/src/unitree_guide/unitree_navigation/maps/sim_map
```

5. launch navigation file -

```
roslaunch unitree_navigation my_navigation.launch
```

Figure 15: Terminal commands to generate map of simulation environment

```
1. launch manipulator controller -
    roslaunch open_manipulator_controller open_manipulator_controller.launch use_platform:=false
2. run manipulator gui contoller -
    roslaunch open_manipulator_control_gui open_manipulator_control_gui.launch
```

Figure 16: Terminal commands to launch the manipulator controller for manipulation

- After Launch the simulation world and go1 controller (Fig.15) the quadruped should be switched from the FixedStand to Trotting to control the translation of the robot.
- The robot first uses its LIDAR sensor along with the Gmapping (SLAM) algorithm to create a map of the simulation environment(Fig.15). This map is essential for navigation as it provides the robot with an understanding of the surroundings, including the positions of obstacles and free spaces in a format that the navigation stack can easily load.

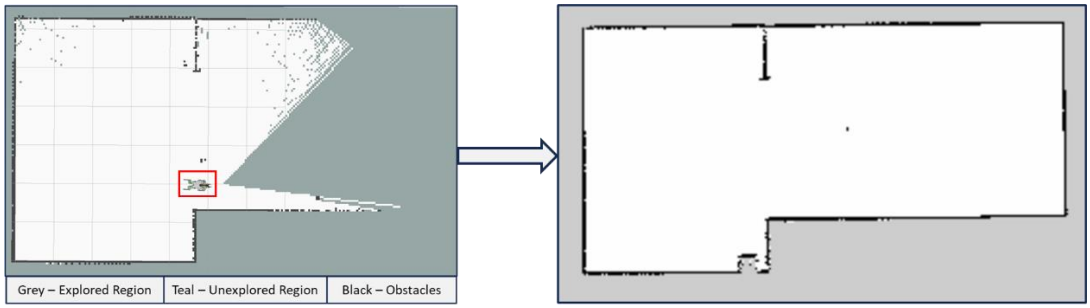

Figure 17: Gmapping of Gazebo environment 1 viewed on Rviz (Left),
Map generated after gmapping saved as a YAML file (Right)

- After the navigation packages are launched the path planning process begins when a target location for the manipulator end effector is provided. This target location is specified as coordinates the manipulator needs to reach through the terminal in the Global frame.

- Potential goal points are generated at a distance of 0.3 meters around the target end effector position. If the planner fails to generate a feasible path, the orientation of the goal point is changed by 90° to check if the new orientation path is feasible.
- Once a feasible goal point and orientation are determined, The quadruped should be switched to move_base mode, a stable and efficient gait for navigation. The previously saved map is loaded into the `move_base` node. The `move_base` node is responsible for combining global planning and local planning to navigate the robot to the target location.
- The Timed Elastic Band (TEB) local planner is used to navigate the quadruped along the global path. If an obstacle is detected along the planned path, the local planner updates the cost map and re-routes the robot to avoid the obstacle while still moving toward the target.
- In RViz, the global and local paths of the robot trajectories are visualized, allowing the operator to monitor the planned and actual paths the robot takes. The laser scan data recorded by the LIDAR is also visualized in RViz, providing real-time feedback on the environment and detected obstacles.

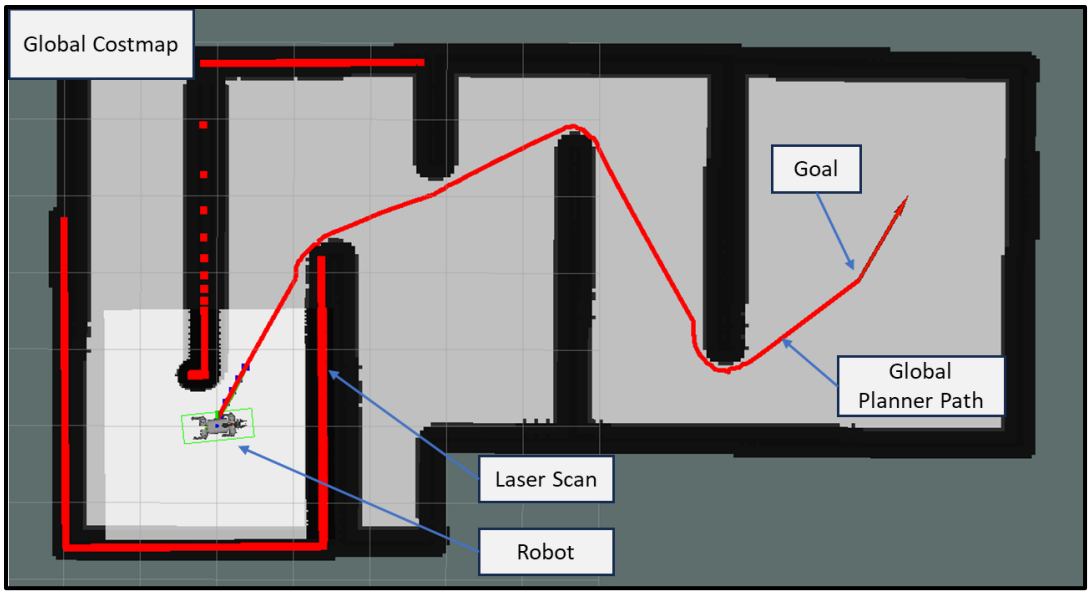

Figure 18: Global Costmap with Laser Scan data, Global Planner trajectory from the Robot position to the Goal position, the rectangular area around the Robot is the Local Costmap, The green box around the Robot is the Go1 footprint area

3.1.3 Manipulator - Motion Planning

Due to the specified tolerance parameters for position and orientation, there might be inaccuracies in the reached target point. To address this, the stereo cameras on the quadruped are used to send accurate 3D positions of the object for the end effector to reach the target point with good precision.

After the quadruped reaches the navigation goal position, the robot's stereo cameras provide accurate 3D coordinates of the object. The accurate 3D coordinates of the object are sent to the manipulator controller in the local frame of the manipulator. The controller computes the Inverse Kinematics of joints to follow the trajectory needed to move the end effector to the specified position accurately. The camera view can be accessed using Image View in RViz by subscribing to the corresponding camera topic.

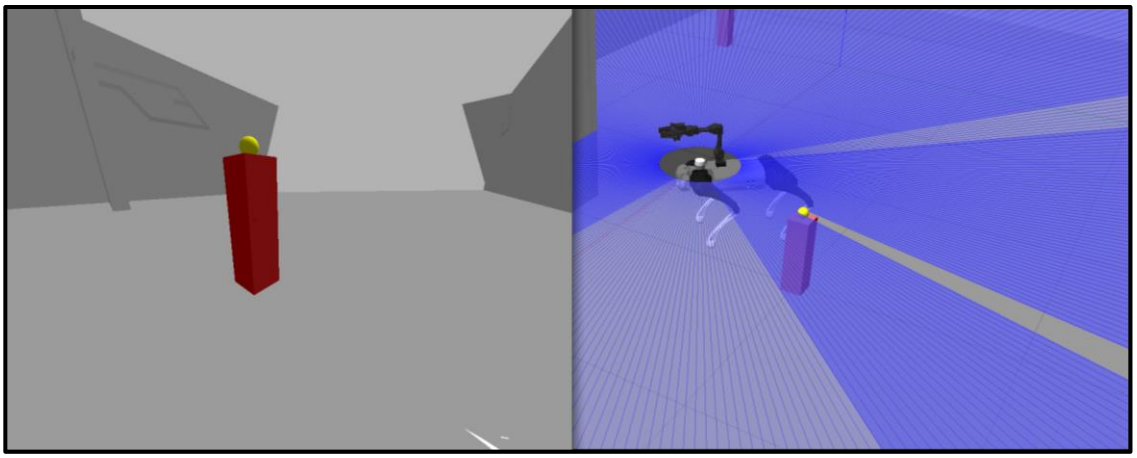

Figure 19: Stereo Camera feedback for effective Manipulator End effector motion planning to object position in the 3D space

3.2 Hardware Setup

This section details the hardware setup for integrating the Unitree Go1 quadruped robot with the Open Manipulator X and the SLAMTEC LiDAR sensor. Unlike in simulation, where it is straightforward to modify the URDF files, integrating these components in hardware requires a more complex setup. This involves using 3D-printed support plates, establishing power connections, and ensuring all components are on the same network for seamless communication. This configuration enables the robot to perform complex tasks efficiently. In the following sections, the Mechanical and Network for communication setups are explained

3.2.1 Mechanical Setup

The SLAMTEC LiDAR and Open Manipulator-X are securely mounted on top of the quadruped using custom 3D-printed support plates. This ensures that all components are firmly attached and positioned optimally for their functions.

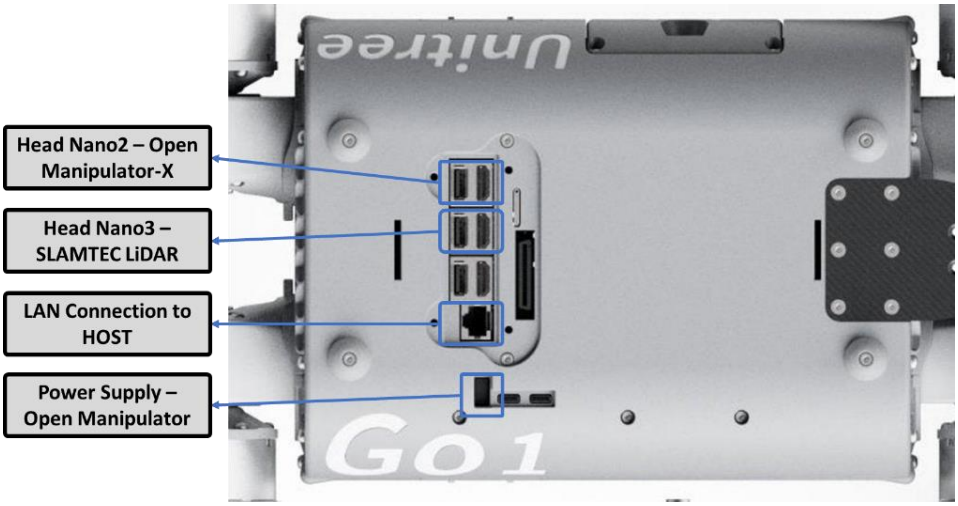

Figure 20: Go1 Mechanical connection setup with the manipulator, LiDAR, and Host PC

- SLAMTEC LiDAR Installation:
    - The LiDAR is mounted to provide an unobstructed 360-degree view of the environment. This positioning is crucial for accurate mapping and obstacle detection.
    - The LiDAR is connected to Head Nano3 via a designated port, which supplies a 5V DC power source.
- Open Manipulator-X Installation:
    - The Open Manipulator-X requires a 12V 5A (60W) power supply to operate efficiently. The quadruped is equipped with a power output of 24V 30A, provided by its rechargeable Lithium-Ion battery.
    - A buck converter is used to step down the voltage from 24V to 12V. The converter is connected to the manipulator, ensuring it receives the appropriate voltage and current for its operation.

- The communication cable for the manipulator is connected to a U2D2 controller, which interfaces with Head Nano2 via the USB port.

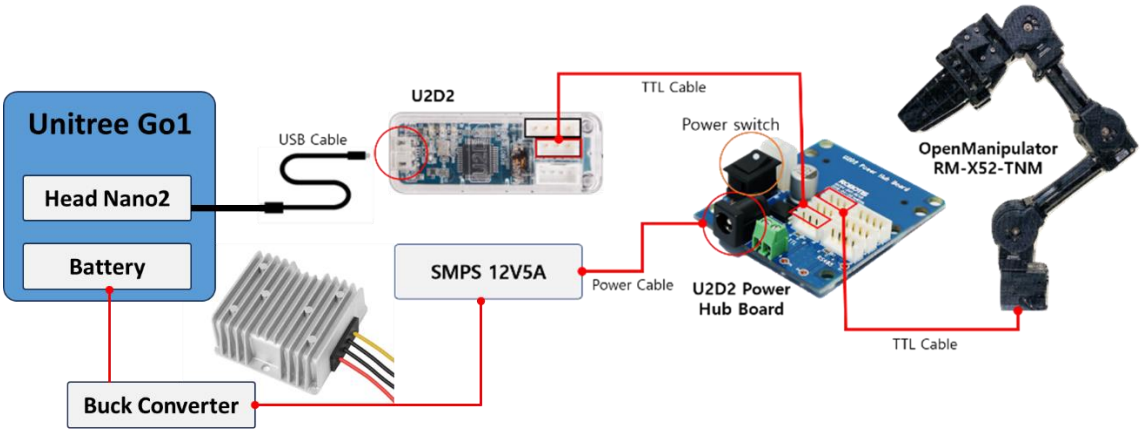

Figure 21: Open Manipulator-X communication and power connection setup with Unitree Go1

3.2.2 Network Setup

The network setup for integrating the Unitree Go1 quadruped robot with the Open Manipulator X and SLAMTEC LiDAR involves ensuring that all components are connected to a unified network, enabling seamless communication between the sensors, manipulators, and control systems. This setup is implemented as a ROS multimachine configuration, with the Host PC serving as the ROS Master. The diagram below illustrates the network configuration:

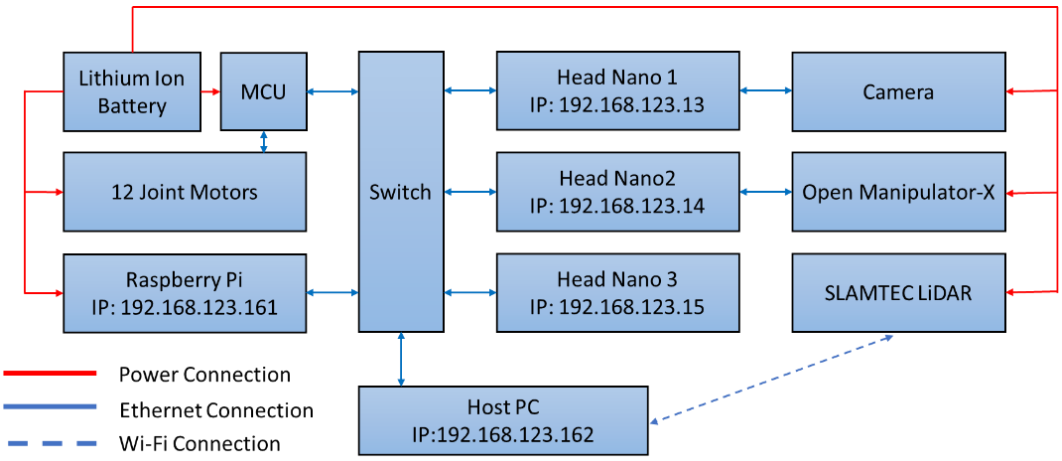

Figure 22: Overview of Manipulator, LiDAR, and Depth camera network setup

- The Host PC acts as the ROS Master, coordinating communication between all ROS nodes running on different devices in the network. The Host PC is connected to Quadruped with a Long Ethernet Cable.
- The LiDAR communicates with the Host PC over Wi-Fi to provide real-time sensor data for navigation and path planning. All the LiDAR packages are hence installed on the host system.
- Open Manipulator-X communication interface U2D2 controller is connected to Head Nano 2. All the ROS packages needed to control the manipulator are installed on the Nano 2 controller.

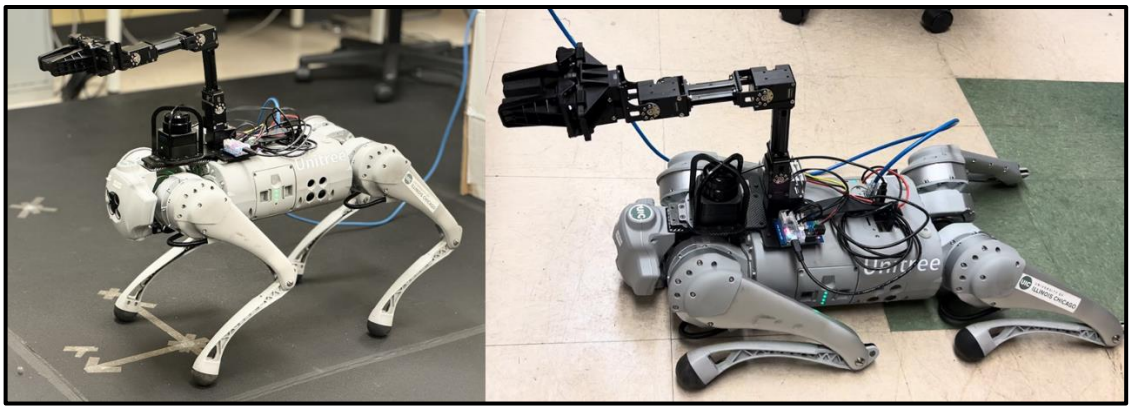

Figure 23: LiDAR, Manipulator, Buck converter, and manipulator controller integrated with quadruped with a 3D-printed base plate

3.2.3 Hardware Testing Environment

To validate the functionality of the integrated system, a closed testing environment was built in the lab.

The first step involved mapping the environment using the SLAMTEC LiDAR. To test the setup, an object is picked up using teleop keyboard controls. The robot drops the object at a specified target location input through the terminal using the saved map. This process verifies the accuracy and reliability of the navigation and manipulation system.

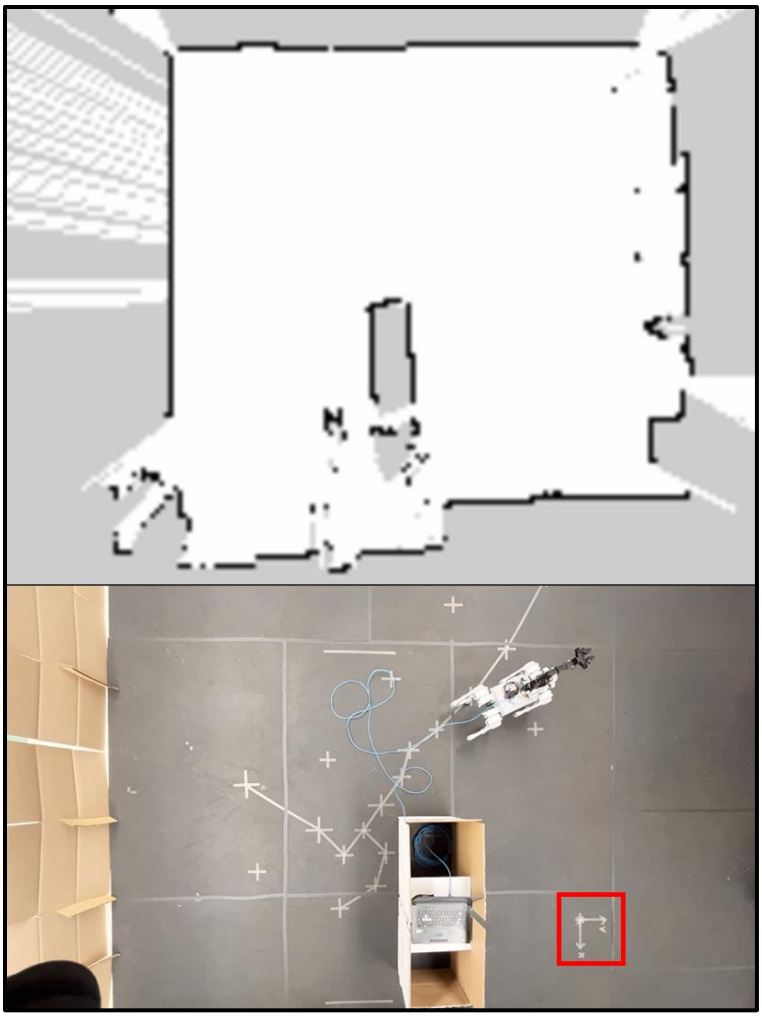

Figure 24: Hardware testing environment map on the (Top), Top View of the hardware testing environment

Figure 25: Front view of hardware testing environment

CHAPTER 4: RESULTS AND CONCLUSION

4.1 Simulation results

The motion planning algorithms were tested in two different Gazebo simulation environments, as described in the Gazebo setup section.

In Gazebo Environment 1, tennis balls were placed at various locations within the simulation world. The locations of these tennis balls, provided as 3D coordinates, were input into the terminal. The integrated quadruped and manipulator system successfully planned paths to the specified locations. Upon reaching each position, the system utilized feedback from the depth cameras to accurately approach and pick up the tennis balls.

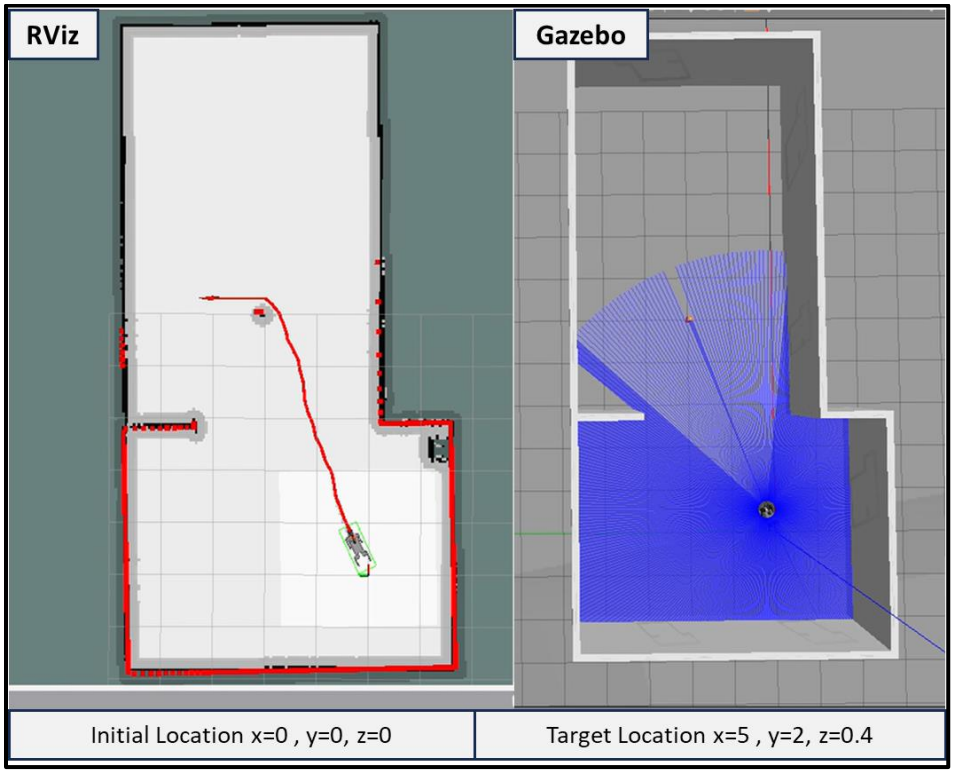

Figure 26: Global Path for the target location viewed on Rviz (Left), Gazebo simulation environment 1 (Right)

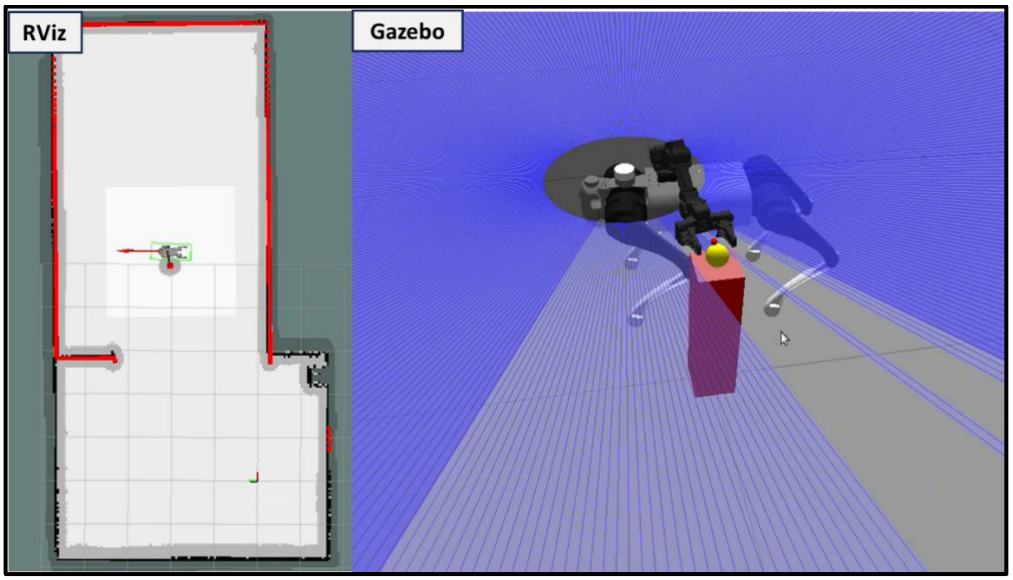

Figure 27: Quadruped reaching the target location as seen in Rviz (Left), Manipulator end effector at the specified target location to pick up the object (Right)

After picking up a ball, the system was given a drop location. The robot then planned and executed a path to this specified drop location.

In Gazebo Environment 2, the process was similar, with tennis ball locations specified as 3D coordinates. However, this environment included obstacles, adding complexity to the path planning and navigation tasks. The integrated system demonstrated its ability to navigate around obstacles while planning paths to the specified ball locations. Upon reaching the positions, it used feedback from the depth cameras to precisely pick up the tennis balls. The system then navigated to the specified drop locations, avoiding obstacles along the way.

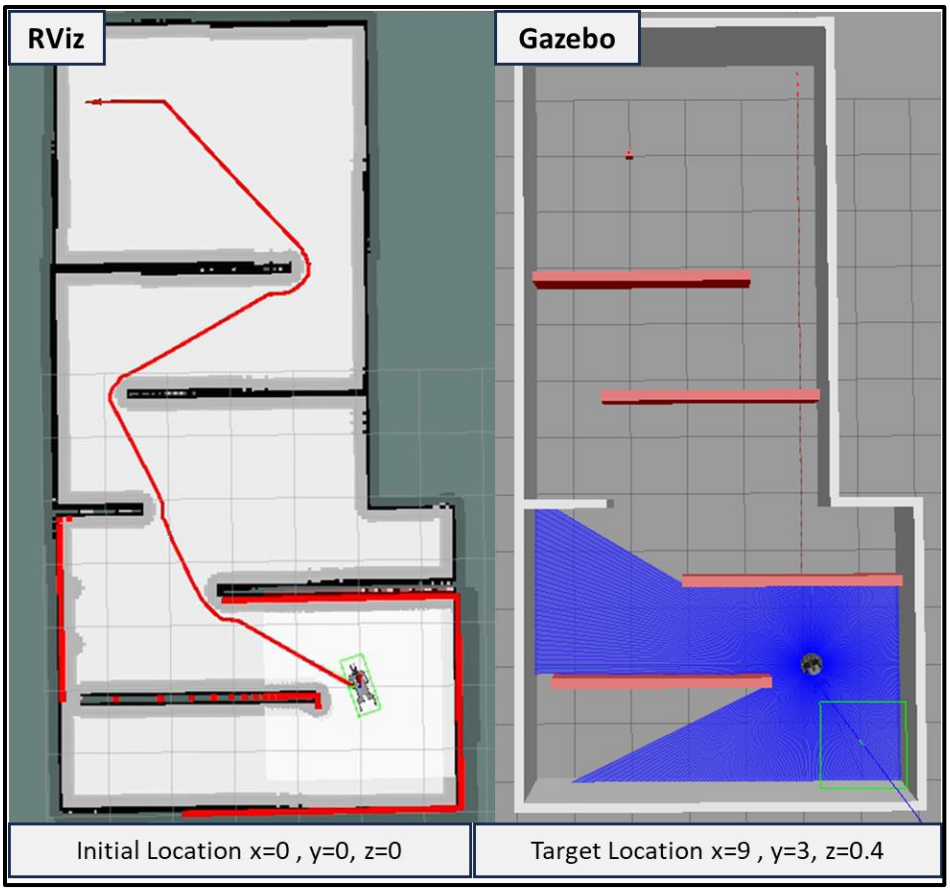

Figure 28: Global Path for the target location viewed on Rviz (Left), Gazebo simulation environment 2 (Right)

The results demonstrate that the integrated system can effectively plan and execute complex motion tasks, including navigating to specific coordinates, using sensor feedback to perform precise manipulations, and transporting objects to designated locations within the simulation environment.

4.1 Hardware results

After configuring the system, we tested the motion planning and control algorithms on the actual hardware. One significant difference between the hardware implementation and the simulation was the absence of depth camera feedback for object pickup. Due to latency issues in the camera's video stream, real-time object detection was not feasible.

To overcome this limitation, we used teleoperated keyboard controls to manually handle object pickup. Once the object was secured, we specified the drop location, and the robot successfully navigated to and released the object at the target location.

One of the most critical metrics for validating the system's performance is repeatability. The object drop test at specified coordinates was conducted 15 times, with successful results in 11 of those trials. To further assess the system's accuracy, a marker was placed on the object to track the robot manipulator's ability to reach the desired end-effector position. Using motion capture, data was collected from the moment the end-effector position was assigned until the manipulator reached the target and released the object.

The recorded data revealed a mean error of 4.56 mm on the x-axis and -21.97 mm on the y-axis when reaching the target, with a standard deviation of 12.58 mm in the x-axis and 14.46 mm in the y-axis. This indicates the system's accuracy in hitting the intended drop points, though some variability exists.

	Mocap_X(mm)	Mocap_Y(mm)	Map_X(mm)	Map_Y(mm)
Test 1	-10.35	-127.56	-12.7	-10.69
Test 2	11.77	-128.25	9.42	-11.38
Test 3	19.31	-158.58	16.96	-41.71
Mean Error			4.56	-21.97
Std Dev.			12.58	14.46

Table 4: Motion capture data for hardware test, Marker Placed on the object picked up and dropped at the target location

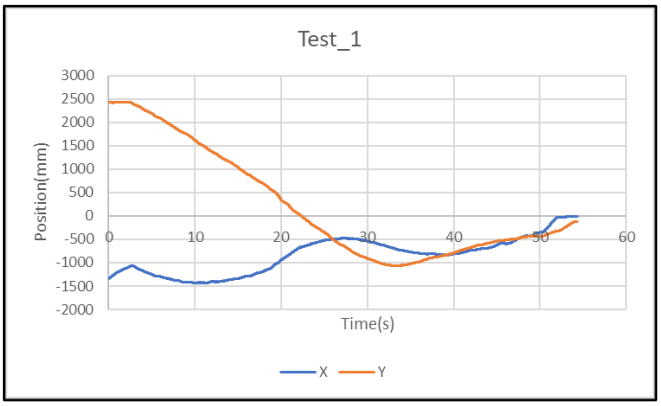

Figure 29:X and Y coordinates data of the pick-up object from the initial position to the target end effector position

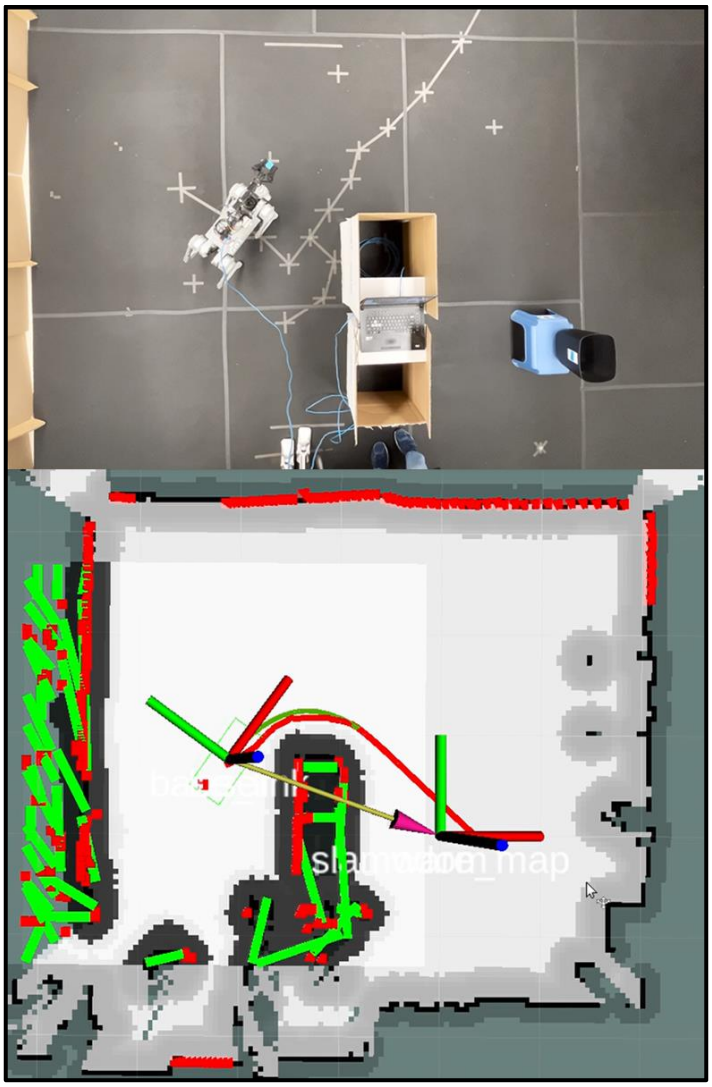

Figure 30: Top view of Quadruped following the Global path indicated by the red path planned towards the target position, The Local path is represented in green. The x-axis of the robot frame (Red) represents the direction of the heading of Go1

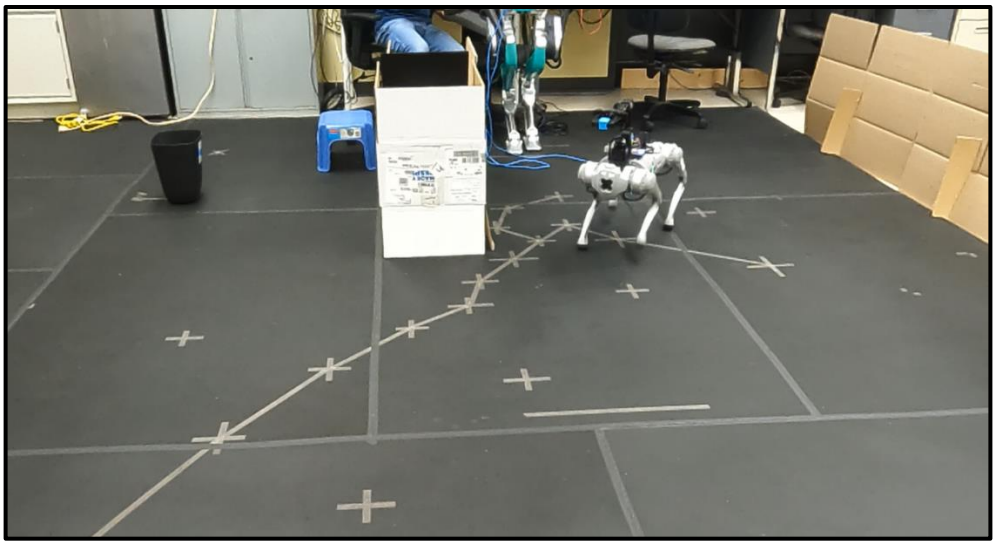

Figure 31: Front view of Quadruped following trajectory to target point carrying a blue color block.

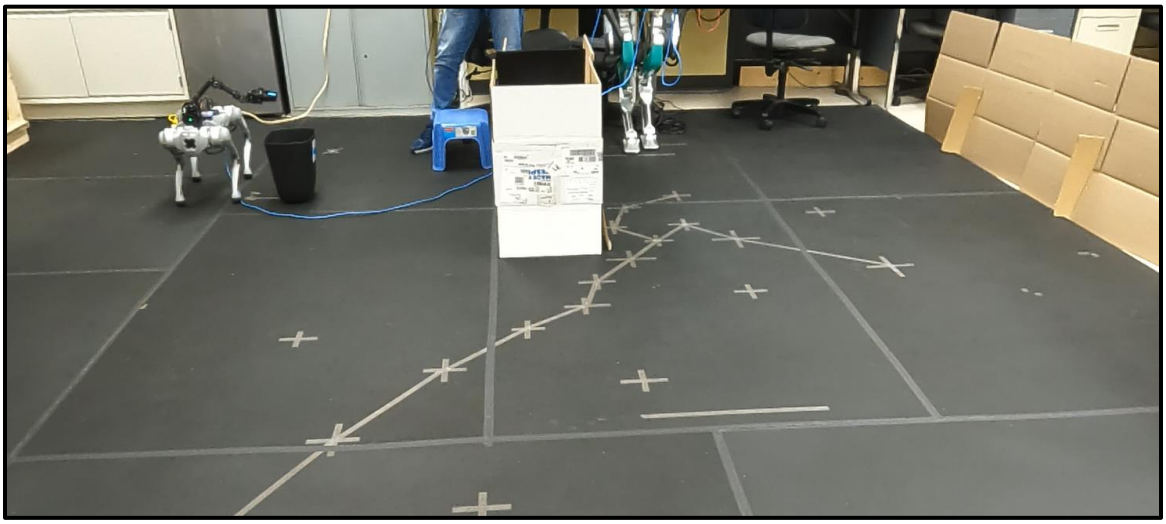

Figure 32: Manipulator dropping the object at the specified target point after navigating to reachable space within the target location

4.1 Conclusion

Building a working quadruped manipulator integrated system was the major goal of this thesis and required a number of critical processes. Initially, a thorough literature review was performed to comprehend the conventional approaches to integration for quadruped

manipulators. The assessment emphasized that the most challenging part of developing a stable and efficient quadruped manipulator is motion planning. Based on existing knowledge about motion planning in quadrupedal robots with appendages, two primary approaches were identified: breaking the system down into its component parts or evaluating it as a whole.

After a thorough analysis of the application, it was decided that the robotic arm would be best used for motion planning as an autonomous system. Consequently, a distinct system strategy was implemented using the ROS packages, which included quadruped route planning and manipulator inverse kinematics (IK).

Simulations were conducted using RViz and Gazebo to validate the quadruped arm system for SLAM path planning. Following positive results from the simulations, the hardware was tested. Although the Go1's depth cameras were unable to perform real-time object tracking due to latency concerns, the successful demonstration of path planning to a defined target point incorporated both the quadruped's navigation and the manipulator's IK motion planning.

REFERENCES

1. Bruno Siciliano and Oussama Khatib. Springer Handbook of Robotics. Springer Berlin Heidelberg, 1 edition, 5 2008. doi:10.1007/978-3-540-30301-5.

2. Priyaranjan Biswal and Prases K. Mohanty. Development of quadruped walking robots: A review. Ain Shams Engineering Journal, 12:2017–2031, 6 2021. doi:10.1016/J.ASEJ. 2020.11.005.

3. Navvab Kashiri et al. Centauro: A hybrid locomotion and high power resilient manipulation platform. IEEE Robotics and Automation Letters, 4(2):1595–1602, 2019. doi:10.1109/ LRA.2019.2896758

4. Hui Chai et al. A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach. Biomimetic Intelligence and Robotics, 2(1):100029, 2022. doi:10.1016/J.BIROB.2021.100029.

5. Dan Zhang, Chao Cheng, Jun Fu, Hang Su, and Luquan Ren. Recent advancements in agriculture robots: Benefits and challenges. Machines 2023, Vol. 11, Page 48, 11:48, 1 2023. doi:10. 3390/MACHINES11010048.

6. Unitree Robotics. Aliengo, 2020. Accessed on Dec. 28, 2022. URL: https://shop. unitree.com/products/aliengo.

7. Maria S. Lopes, A.Paulo Moreira, Manuel Silva, and Filipe Santos. A Review on Quadruped Manipulators. Unpublished, September 2023.

8. Maria S. Lopes, A. Paulo Moreira, Manuel Silva, and Filipe Santos. Robotic arm development for a quadruped robot. Unpublished, October 2023.

9. Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. Robotics: Modelling, Planning and Control. Springer London, 2009. doi:10.1007/978-1-84628-642-1

10. Kevin (Kevin M.) Lynch and Frank C. Park. Modern robotics : mechanics, planning, and control. Cambridge University Press, 2017.

11. Mark W. Spong, Seth Hutchinson, and Vidyasagar M. Robot Modeling and Control, volume 141. Wiley, 1st edition, 2005.

12. Robotic Systems Control. Dynamics of robotic manipulators, 2022. Accessed on Mar. 25, 2023. URL: https://www.youtube.com/watch?v=QN-Awth50aA.

13. Robotic Systems Lab: Legged Robotics at ETH Zürich. Anymal using an elevator, 2017. Accessed on Dec. 28, 2022. URL: https://www.youtube.com/watch?v=gM1z60aeunU.

14. Boston Dynamics. Spot's on it, 2021. Accessed on Dec. 28, 2022. URL: https://www. youtube.com/watch?v=7atZfX85nd4.

15. Deep Robotics. Jueying x20, 2022. Accessed on Dec. 28, 2022. URL: https://www. deeprobotics.cn/en/products.html.

16. Unitree Robotics. Unitree robotics aliengo + z1 for fire rescue, 2022. Accessed on Dec. 28, 2022. URL: https://www.youtube.com/watch?v=7CVwGY65In8

17. Thushara Sandakalum and Marcelo H. Ang. Motion planning for mobile manipulators—a systematic review. Machines, 10, 2 2022. doi:10.3390/MACHINES10020097.

18. Jun Li et al. Whole-body control for a torque-controlled legged mobile manipulator. Actuators, 11:304, 10 2022. doi:10.3390/ACT11110304.

19. Unitree Robotics. Ros simulation packages for unitree robots, 2020. Accessed on Dec. 22, 2022. URL: https://github.com/unitreerobotics/unitree_ros.

20. Chitta et al. ros_control: A generic and simple control framework for ros. The Journal of Open Source Software, 2017. doi:10.21105/joss.00456.

21. Guiyang Xin, Fanlian Zeng, and Kairong Qin. Loco-manipulation control for arm-mounted quadruped robots: Dynamic and kinematic strategies. Machines, 10:719, 8 2022. doi:10. 3390/MACHINES10080719/S1.