NAVIGATION AND CONTROL OF COOPERATIVE MOBILE ROBOTS: B.TECH PROJECT REPORT SUBMITTED TO IIT TIRUPATI

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

MECHANICAL ENGINEERING

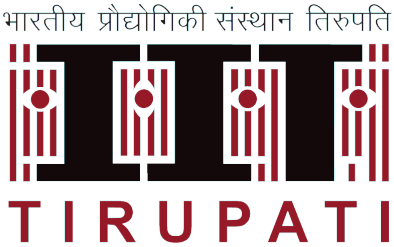
by

SUMUKH PORWAL ME20B041

RITWIK DAS ME20B034

Supervisor

Dr. Thiyagarajan R



DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

MAY 2024

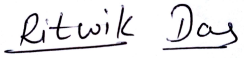
DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/- data/fact/source in our submission to the best of our knowledge. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 08-05-2024

Signature
Sumukh Porwal
ME20B041

Place: Tirupati
Date: 08-05-2024



Signature
Ritwik Das
ME20B034

BONA FIDE CERTIFICATE

This is to certify that the report titled **NAVIGATION AND CONTROL OF CO-OPERATIVE MOBILE ROBOTS: B.TECH. PROJECT REPORT**, submitted by **Sumukh Porwal and Ritwik Das**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the project work done by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Tirupati
Date: 08-05-2024

Thiyagarajan R
Supervisor
Assistant Professor
Department of Mechanical Engineering
IIT Tirupati - 517619

ACKNOWLEDGMENTS

Thanks to Dr. Thiyagarajan R, for his mentorship and guidance throughout the Project and also for all the patience he put into correcting the mistakes and suggesting the necessary changes which made our journey more smooth. We might likewise want to thank the BTP panel for the fruitful discussion we had during the presentations. Further, we would like to thank the Department of Mechanical Engineering for providing the funds to see the project till its completion.

ABSTRACT

KEYWORDS: Mobile robot, Path planning, SLAM, Autonomous Navigation, Holonomic Drive, Multi-Robot System, Cooperative Navigation

This project proposes a comprehensive investigation into the navigation of three-wheeled omnidirectional robots, aiming to enhance their manoeuvrability in dynamic and complex environments, making them suitable for tasks requiring precise and agile movements.

The project thoroughly examines the robots' hardware and kinematics, focusing on incorporating omnidirectional wheel features into navigation algorithms. Simulations provide controlled environments to validate and optimize cooperative navigation algorithms, considering scenarios like obstacle avoidance and dynamic changes. Integrating LiDAR enhances the robots' perception capabilities. Path planning algorithms to optimize trajectories for efficiency and collision avoidance, adapting in real-time to dynamic obstacles. Physical experiments with the robot validate the system's performance and durability.

This project also focuses on developing a multi-agent system that enables seamless communication and coordination among the robots to achieve common navigation objectives. Communication and coordination protocols will be developed through ROS to facilitate seamless information exchange among the robots, employing consensus algorithms and task allocation strategies for effective collaboration. The project aims to contribute to cooperative robotics advancements, finding applications in industrial automation, logistics, and collaborative research environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ROS	Robotics Operating System
SLAM	Simultaneous Localization and Mapping
AMCL	Adaptive Monte Carlo Localisation
MRS	Multi Robot System
PID	Proportional Integral Derivative
LiDAR	Light Detection and Ranging
URDF	Unified Robot Description Format
Xacro	XML macro language

NOTATION

u	velocity along x in body frame, $\frac{m}{s}$
v	velocity along y in body frame, $\frac{m}{s}$
r	angular velocity along z in body frame, s^{-1}
ψ	orientation of body frame w.r.t. inertial frame in degrees
θ	orientation of wheel frame w.r.t body frame in degrees
ϕ	Complimentary of angle between wheel axis and axis of passive roller in degrees
J	Jacobian matrix
η	position vector of robot in inertial frame
ζ	position vector of robot in body frame
W	input configuration matrix
K	inverse matrix of W
d_{xi}	x co-ordinate of i^{th} wheel frame w.r.t. body frame
d_{yi}	y co-ordinate of i^{th} wheel frame w.r.t. body frame

CHAPTER 1

INTRODUCTION

The contemporary industrial landscape necessitates the efficient handling of tasks. However, the inflexibility of these traditional systems poses challenges for adapting to evolving tasks, leading to substantial costs in redesigning workfloors. In contrast, the inherent adaptability of robots positions them as a versatile solution to this predicament. While a single robot could replace traditional machinery, its complexity and costliness, coupled with vulnerability to unit failures, hinder seamless task execution.

This project advocates for the adoption of Multi-Robot Systems (MRS) as a strategic alternative, aiming to address both complexity and cost-effectiveness. MRS, characterized by cooperative behavior among structurally and functionally similar yet task-specialized robots, emerges as a pragmatic solution. Precision in mobile robots, particularly crucial for cooperative tasks, is achieved through the integration of laser sensor guidance, mitigating errors and ensuring the safe and accurate functioning.

Recognizing the significance of maneuverability in transportation-centric robotic systems, the project introduces omnidirectional wheels. This innovative approach allows robots within the MRS to move freely in any direction, enhancing system resilience. The amalgamation of robots with omnidirectional wheels, operating in tandem with vision guidance, emerges as a feasible and potent solution for the intricate challenge of transporting heavy loads in dynamic industrial environments. In essence, the project aims to pioneer advancements in the realm of multi-robotic systems, contributing to the paradigm shift in industrial load-handling practices.

1.1 Motivation

In recent years, the trend of multi-mobile manipulator systems has gained significant traction in the field of robotics, captivating the attention of researchers and practitioners alike. The growing interest stems from the potential of these systems to revolutionize

various applications, ranging from industrial automation to search and rescue missions. Researchers worldwide are actively exploring the integration of cooperative omnidirectional mobile robots as the foundation for advanced multi-mobile manipulator setups. This trend is marked by an increased emphasis on enhancing collaborative capabilities, adaptability, and efficiency in dynamic environments. The pursuit of innovative solutions to enable seamless cooperation among mobile platforms and manipulators underscores a collective effort to address complex real-world challenges. As this paradigm continues to evolve, the project's motivation lies in contributing to this cutting-edge trend, anticipating breakthroughs in the development of cooperative robotic systems for a diverse array of applications.

1.2 Objective

The primary objective of this project is to design, develop, and implement a cooperative omnidirectional mobile robot system. Recognizing the expansive nature of the broader trend in multi-mobile manipulator systems, our focus will be on establishing a solid foundation by creating a versatile and efficient cooperative platform. Within the project's timeframe, our aim is to engineer a modular system capable of seamless collaboration among omnidirectional mobile robots. This includes the integration of advanced sensor networks, communication protocols, and cooperative control algorithms. By prioritizing the development of a robust and scalable cooperative mobile robot system, we position ourselves strategically within the overarching trend, paving the way for future expansion into multi-mobile manipulator configurations. This objective aligns with our commitment to incremental progress, ensuring a realistic and impactful contribution to the evolving landscape of cooperative robotics within the designated time frame.

1.3 Multi robot systems

Multi Robot Systems are defined as a system of multiple robots which can cooperate and communicate with each other to accomplish certain tasks. The fact that single robots are limited in terms of capabilities and efficiency led to the development of Multi Robot Systems. Multi robot systems are flexible and operate with increased efficiency. In this

project a system of three robots with cooperative behaviour is presented.

1.4 Co-operative behaviour

It is highly essential for the robots to work together simultaneously without robots' collision. Cooperative robots are mainly defined as those that exhibit joint behaviour that is directed towards a specific goal. This helps robots to work as a group. One advantage of the co-operation is that when a robot is malfunctioned, the other robots distribute the task efficiently. In practice, a robot's end effector might deviate from the desired trajectory. In this project, the three robots work towards a common goal from the start position to the target position. The scope of this project limits to developing control system to accomplish the objectives using ROS, demonstrating its application through Simulation and make a prototype of the proposed cooperative omnidirectional mobile robots system.

CHAPTER 2

Literature Review

Cooperative navigation of mobile robots involves the collaborative efforts of multiple robotic agents to achieve shared goals in navigating and mapping unknown environments. This approach addresses challenges such as localization errors, occlusions, and scalability, making it crucial for surveillance, exploration, and disaster response applications. Researchers have explored various techniques to enhance the efficiency and robustness of cooperative navigation.

Cooperative Simultaneous Localization and Mapping (SLAM) advancements have gained attention. Tao *et al.* (2010) presented a collaborative SLAM algorithm that leverages inter-robot communication to refine individual maps and achieve global consistency. Multi-robot exploration strategies have also been explored, as evidenced by the work of Leung *et al.* (2010), who introduced a scalable approach for coordinating robot teams in unknown environments.

The design of omnidirectional wheels empowers these robots with exceptional manoeuvrability. Key kinematic models, such as the differential drive model, have been adapted and extended to capture the unique features of 3-wheeled configurations. Siegwart *et al.* (2011) have explored forward and inverse kinematics of both holonomic and non-holonomic constraints. Few have also considered factors like wheel slippage and motor limitations to refine the accuracy of kinematic models in real-world scenarios.

Control strategies, including trajectory planning and feedback control, are crucial in optimizing the performance of 3-wheeled omnidirectional robots. Trajectory planning algorithms, encompassing classical and AI-based approaches, enable efficient navigation in complex environments. Feedback control mechanisms, such as PID controllers, contribute to stability and precision in motion regulation.

AMCL is a widely used algorithm for robot localization in environments with uncertainty. It employs a particle filter approach, where a set of particles represents the possible locations of a robot. These particles are adaptively updated based on sensor

measurements, allowing the robot to estimate its pose even in dynamic environments. dos Reis *et al.* (2019). introduced AMCL into the ROS environment, emphasizing its adaptability to changing conditions and effectiveness in mitigating the challenges of real-world localization.

SLAM is a fundamental problem in robotics, aiming to enable a robot to build an environment map while simultaneously localizing itself within that map as presented by Durrant-Whyte and Bailey (2006). Additionally, LiDAR-based SLAM has gained prominence, offering a robust solution for 3D mapping and localization using LiDAR sensors Debeunne and Vivet (2020).

Integrating AMCL and SLAM is crucial for achieving robust and accurate localization in dynamic environments. Combining the adaptability of AMCL with the mapping capabilities of SLAM provides a comprehensive solution for simultaneous localization and mapping tasks. Recent work by Zhang *et al.* (2013) proposed an integrated approach, demonstrating the fusion of particle filter localization and mapping techniques for improved performance in challenging scenarios.

Applications of cooperative navigation robots involving various strategies, including distributed sensing, communication, artificial potential fields, and collaborative SLAM, spanning diverse domains, from warehouse automation to service robotics. The adaptability of their kinematics to tight spaces and dynamic environments positions these multi-robot systems as a valuable asset in industries requiring agile and precise mobility.

Cooperative navigation has several strategies. The formation of multiple robots based on the leader-follower mechanism has been explored by Qian and Xi (2018), where some numerical simulations are displayed to verify the feasibility and effectiveness of the designed controller and observer. Distributed multi-robot systems are explored by Ota (2006), going into the current trends and the potentials of multi agent robotics systems.

CHAPTER 3

NAVIGATION AND CONTROL OF OMNI-DIRECTIONAL ROBOTS

3.1 About this Chapter

In this chapter, the report presents the flow of work done to accomplish the cooperation behaviour using the vision guidance for pick and place applications with detailed mathematical models, assumptions and simulation considerations.

- Mathematical formulation for closed loop forward kinematics.
- How the motion is made Omni-directional?
- Mathematical formulation for closed loop inverse kinematics.
- What kind of experimental setup will be used for testing?
- Materials used for the prototype.
- Process flow of implementing SLAM and Autonomous Navigation
- Design considerations of prototype.
- Simulations in GAZEBO for visualization.
- Working of ROS2 and its implementation
- Hardware Implementation
- Cooperative Navigation

3.2 Mathematical Formulation

Kinematics of a mobile robotic system can be formulated in two ways.

Forward Differential Kinematics:

For a given set of angular velocities as input commands, finding the derivatives of generalized co-ordinates/finding system's motion is called Forward Differential Kinematics.

Inverse Differential Kinematics:

For a desired/given derivatives of generalized coordinates/ given position trajectory, finding the corresponding angular velocities is called Inverse Differential Kinematics.

3.2.1 Forward Differential Kinematics

Velocities in an inertial frame in terms of body frame velocities.

Let u, v, r be x-axis velocity, y-axis velocity, and the angular velocity of the robot with respect to the body frame, whereas x, y, and ψ be the pose of body frame with respect to the global inertial frame. We know that the velocities can be equated when the system is static at an instantaneous point. This gives us the forward kinematic equation.

$$\dot{\eta} = J(\psi) * \zeta \tag{3.1}$$

Where,

- $\dot{\eta}$ is the velocity vector with respect to the inertial frame.
- ζ is the velocity vector with respect to the body frame.

Jacobian J(ψ) is given by:

$$J(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Where [ψ] is the orientation of body frame w.r.t. global inertial frame.

ζ can be calculated from the angular velocities of the wheels. The relation between the wheel angular velocities and ζ depends on the type of wheels and the configuration in which they are arranged on the robot. A generalized dependency of ζ on angular velocities of the wheels is given by,

$$\zeta = W * \omega \tag{3.3}$$

Where,

- W is the wheel input matrix/input configuration matrix.
- ω is the angular velocity vector of the wheels.

Wheel input matrix or input configuration matrix, W is defined as:

$$W = K^{-1} \tag{3.4}$$

as K is square matrix in our case. Each row of K is given by,

$$K_i = \begin{bmatrix} \frac{1}{a_i} & \frac{tan(\phi_i}{a_i} \end{bmatrix} \begin{bmatrix} cos(\theta_{Bi}) & sin(\theta_{Bi}) \\ -sin(\theta_{Bi}) & cos(\theta_{Bi}) \end{bmatrix} \begin{bmatrix} 1 & 0 & -d_{yi} \\ 0 & 1 & d_{xi} \end{bmatrix} \tag{3.5}$$

Where,

- a_i is the radius of i_{th} wheel.
- ϕ_i is the complimentary angle of the angle between wheel axis and axis of passive rollers on the i_{th} wheel.
- θ_{Bi} is the orientation of the frame attached to i_{th} wheel w.r.t. body frame.
- (d_{xi}, d_{yi}) are the coordinates of origin of the frame attached to the i_{th} wheel w.r.t. body frame.

If there are n wheels, the size of K will be n×3. Here we use Omni-directional wheels in triangular configuration. So K is a 3×3 matrix.

For our Omni-directional tri-wheeled robot:

- $\phi_i = 0$
- $\theta_{B1} = \frac{\pi}{2}$
- $\theta_{B2} = \frac{7\pi}{6}$
- $\theta_{B3} = \frac{11\pi}{6}$

If a is the wheel radius and l is the distance from the center of the robot(i.e. the origin of body frame) to the wheel center and substituting the above values in the expression for K, we get the input configuration matrix W as,

$$W = \frac{a}{3} \begin{bmatrix} 0 & -\sqrt{3} & \sqrt{3} \\ 2 & -1 & -1 \\ \frac{1}{l} & \frac{1}{l} & \frac{1}{l} \end{bmatrix} \tag{3.6}$$

Angular velocity vector omega is represented as:

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3.7}$$

Euler method to find robot's position

To get the position of the robot at any time, we perform numerical integration using Euler's rule which is given by,

$$\frac{dx(t)}{dt} = \dot{x}_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \tag{3.8}$$

Thus we get,

$$[\eta]_{i+1} = J(\psi)[\dot{\zeta}]dt + [\eta]_i \tag{3.9}$$

Therefore, given the angular velocities of the wheels, we can find velocity and position of the robot using this forward kinematics model.

3.2.2 Inverse Differential Kinematics

Having formulated the mathematical equations for forward kinematics and both Jacobian $J(\psi)$ and wheel input matrix W being invertible, formulating inverse kinematics equations is just a matter of inverting the forward kinematics equations (3.1) and (3.3). Given the desired ground frame velocity vector η of the robot, the corresponding body frame velocity vector ζ can be found by using the equation,

$$\zeta = J(\psi)^{-1} * \dot{\eta} \tag{3.10}$$

Using this body frame velocity vector, the required set of angular velocities to generate that motion is calculated using the equation,

$$\omega = W^{-1} * \zeta \tag{3.11}$$

3.3 Experimental Setup

Experimental setup consists of a remote PC and a robot which captures the working environment that has obstacles in it. It sends this information to the remote PC where GUI for navigation is running. Same information is used by onboard micro processor where path planning and navigation control takes place. Now, this information is fed as an input to the mobile robot actuators for navigating to the destination point with obstacle avoidance. The communication between the remote PC and the robotic system is wireless.

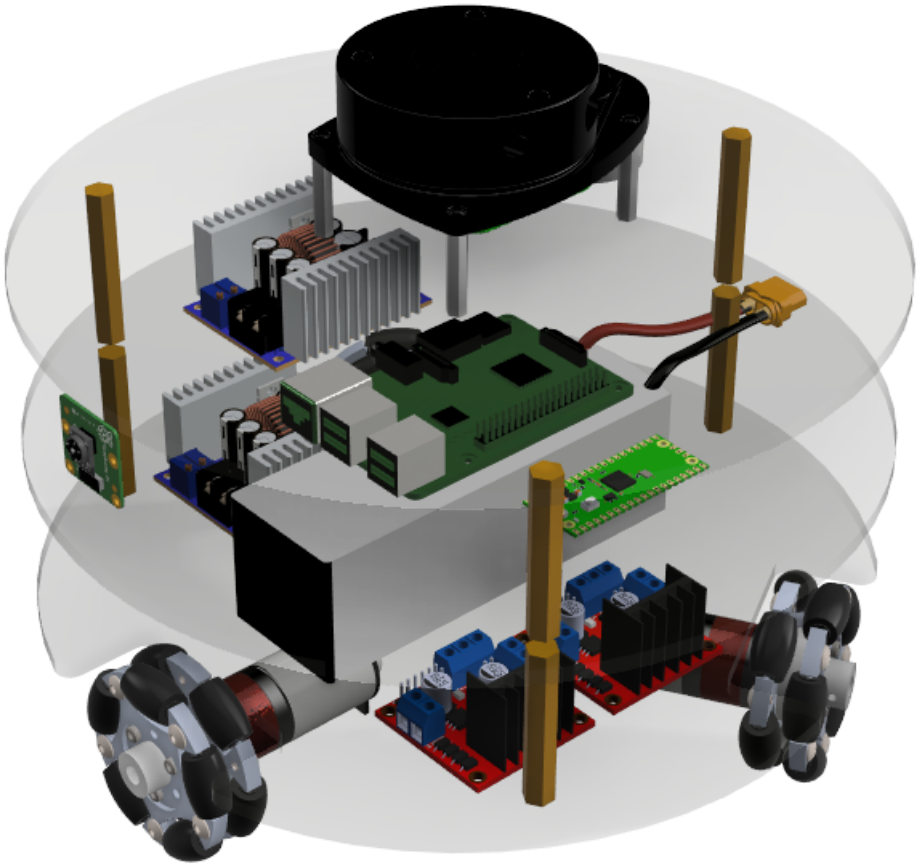


Figure 3.1: Robot

3.4 Materials Used

Part	Quantity
Omni wheels	6
Motors	6
Acrylic chassis	6
Standoffs	12
Micro controller	2
Motor drivers	4
Micro processor	2
DC-to-DC converter	4
Battery	2
RP LiDAR	2

Table 3.1: Materials Used

3.5 Procedure, Techniques and Methodologies

3.5.1 Simulation and Visualization

Simulation is used in the development and testing of robot, providing a controlled environment to validate algorithms and behaviors.

The simulation begins with the creation of a detailed CAD model representing the physical structure of the 3-wheeled omnidirectional robot. This model includes information about the geometry, dimensions, and configuration of the robot's components, such as the wheels and chassis.From the CAD model, URDF (Unified Robot Description Format) and SDF (Simulation Description Format) files are generated. These files define the robot's links, joints, and physical properties required for accurate simulation in Gazebo. The URDF file encapsulates the robot's kinematics, while the SDF file specifies simulation-specific details.

The generated URDF and SDF files are then used to spawn the 3-wheeled omnidirectional robot in the Gazebo simulation environment. Gazebo provides realistic physics

simulations and enables the robot to interact with its surroundings in a simulated 3D space. By embedding the Nav2 stack into the simulation environment, this integration serves as the neural network of the simulation, connecting the simulated robot with navigation algorithms.

Utilizing Gazebo and the Nav2 stack, we test and validate SLAM algorithms in the simulated environment. The robot explores its surroundings, generating a map and accurately localizing itself within that map. The integrated Nav2 stack facilitates the testing of navigation algorithms. The robot is provided with high-level goals, and the stack computes the trajectory, ensuring the robot navigates through the simulated environment while avoiding obstacles.

In the simulation environment, various navigation parameters, such as those related to the costmap, planner, and recovery behaviors, can be tuned to optimize the robot's performance. This tuning process ensures adaptability to different environments and enhances the robot's efficiency. RViz is utilized for real-time visualization of the simulation data. The robot's pose, generated maps, and planned trajectories are visualized, providing insights into the performance of the Nav2 stack in the simulated environment.

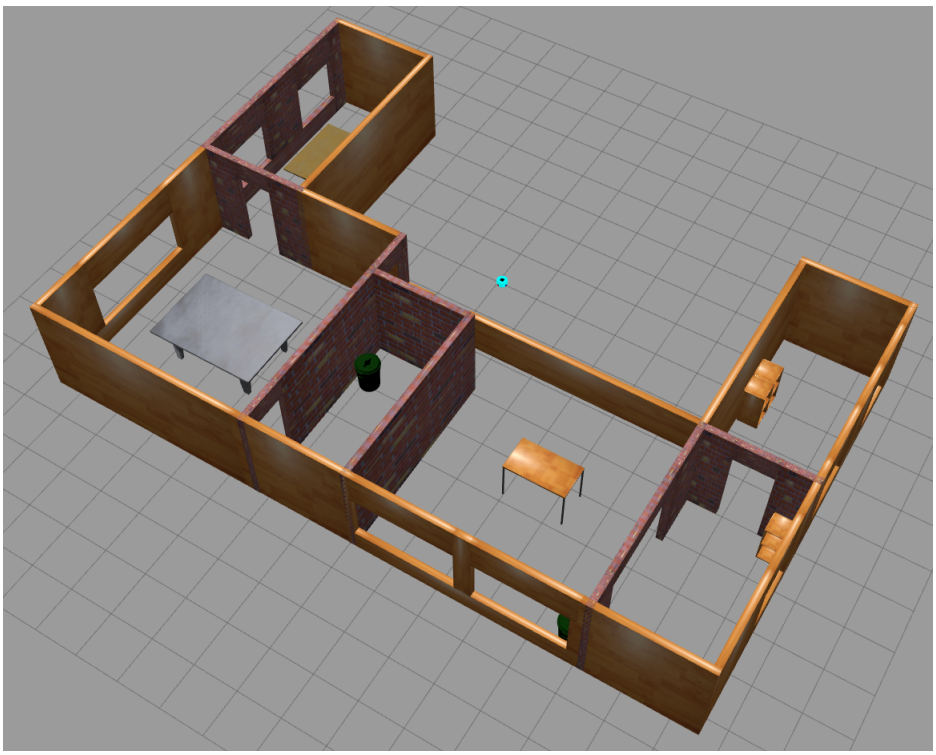


Figure 3.2: Virtual Environment

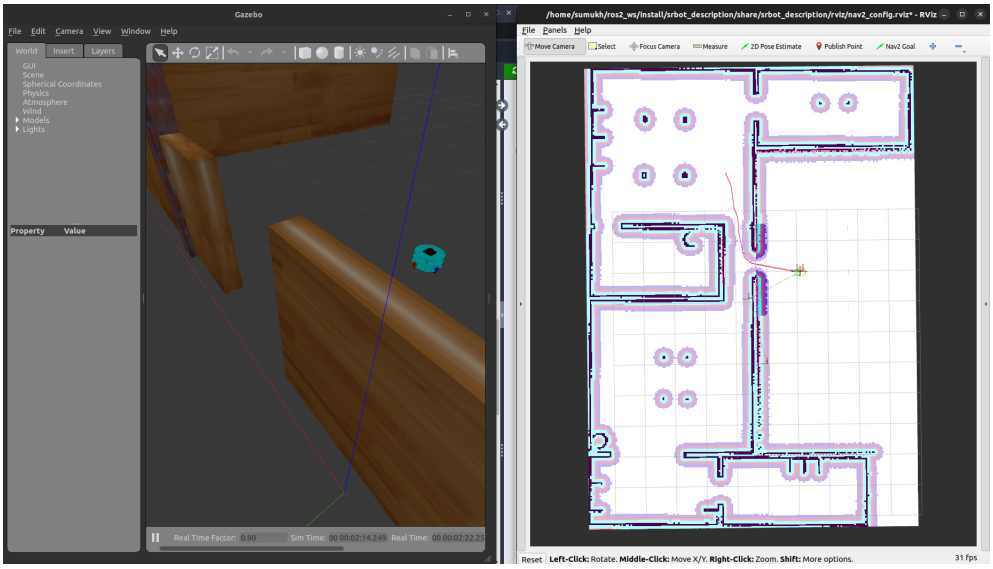


Figure 3.3: Robot Simulation

3.5.2 Pose Estimation

Odometry using motor encoders is used to estimate the robot's position and orientation by tracking the rotation of its wheels. Motor encoders are sensors attached to the robot's motors that measure the number of rotations or angular displacements, allowing for the calculation of the distance traveled and changes in orientation. This odometry information is invaluable for navigation and control systems, enabling robots to maintain an estimate of their pose (position and orientation) as they move through an environment.

$$\omega_i = \frac{n_t - n_{t-1}}{\Delta t * cpr_i} \tag{3.12}$$

where ω_i is the angular speed of the motor (rad/s), n_t is current value of the encoder ticks of the motor and n_{t-1} is value of the encoder ticks of the motor for the last iteration, Δt is the time difference between the iterations, and cpr_i number of encoder ticks per evolution of that motor.

The value of the encoder ticks were measured using interrupts in the controller program.

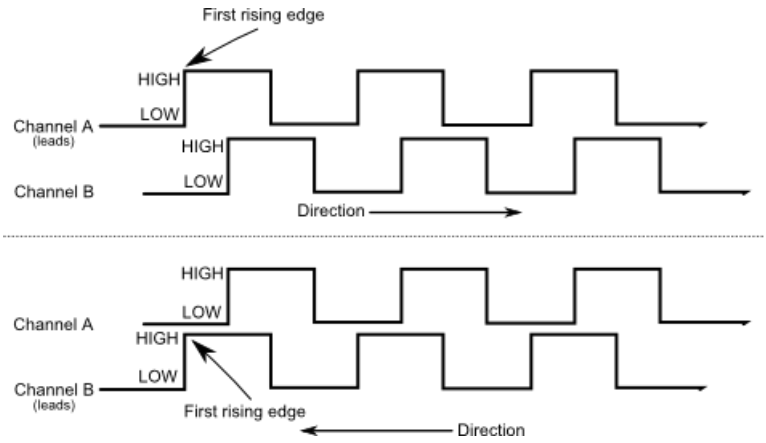


Figure 3.4: Calculating angular velocity of motor using encoders

Motor encoders provide a direct measurement of the rotational movement of a robot's wheels. By counting the number of encoder pulses, which correspond to wheel rotations, the system can determine the distance traveled and the change in orientation. By knowing the wheel circumference and the number of encoder pulses, the angular velocity of each wheel is calculated then the system does Forward Kinematics (3.2.1) to estimate the robot's translational displacement and changes in the robot's orientation. Odometry information is integrated over time to estimate the robot's pose. This involves keeping track of the accumulated translational and angular displacements to update the robot's position and orientation in the environment.

While motor encoders provide valuable odometry data, they are prone to errors such as wheel slippage and encoder drift. Advanced odometry systems incorporate error correction mechanisms to compensate for inaccuracies and improve the overall accuracy of pose estimation. Odometry data is fused with information from LiDAR and inertial measurement units (IMU), to enhance the accuracy and robustness of the robot's pose estimation.

3.5.3 Map Representation

Map representation is a crucial aspect of robotic navigation, it involves creating and utilizing maps to enable the robot to understand and navigate its environment. Let's explore the details of map representation and how it is used in Nav2:

Types of Maps:

Occupancy Grid Maps: These maps represent the environment as a grid, where each cell contains information about the occupancy state (occupied, free, or unknown). Occupancy grid maps are commonly used for representing the static structure of the environment.

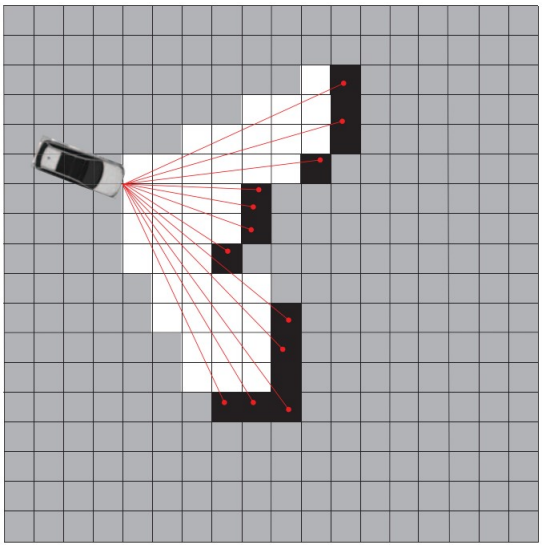


Figure 3.5: Grid cells with a high probability of being occupied are coloured black, and free grid cells are marked with white colour. Grid cells with an unknown state are displayed in grey colour Nuss *et al.* (2016)

Costmaps: Costmaps assign a cost to each cell in the occupancy grid, indicating the traversability of that cell. The costmap is dynamic and can be updated based on sensor data, taking into account not only obstacles but also factors like terrain difficulty.

SLAM (Simultaneous Localization and Mapping):

Simultaneous Localization and Mapping is a technique used to create maps of an environment while the robot simultaneously localizes itself within that environment. SLAM algorithms utilize sensor data, such as LIDAR and camera information, to build a map and determine the robot's position relative to that map. SLAM-generated maps are often used as the foundation for navigation in unknown or partially known environments.

Map modifications

Dynamic Maps: Nav2 employs dynamic maps that can be updated in real-time as the robot moves and gathers new sensor data. This ensures that the map representation is always current, allowing the robot to adapt to changes in the environment.

Costmap Updates: Costmaps, in particular, are continually updated based on sensor information, enabling the robot to react to dynamic obstacles and changes in the environment.

3.5.4 Localization:

Localization in Nav2 is the process by which a robot determines its position within the environment. This is crucial for accurate navigation, especially when the robot needs to follow a predefined path or respond to user-specified goals.

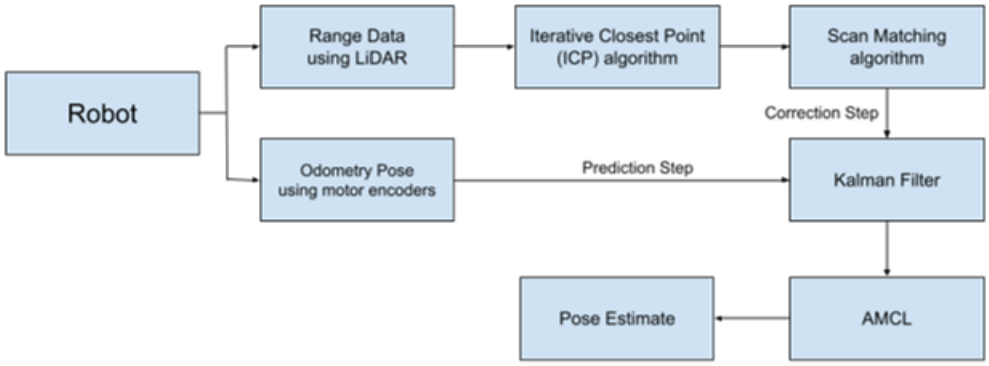


Figure 3.6: Flow Chart of Localisation

Algorithms used for localization of the robot

AMCL (Adaptive Monte Carlo Localization) / Particle Filter:

Monte Carlo Localization (MCL) is a probabilistic approach where the robot's pose is represented by a set of particles, each with a weight indicating its likelihood of being the true pose. Particle filtering helps the robot converge to an accurate estimate of its pose. MCL is effective in handling uncertainty and is suitable for localization in both known and partially unknown environments.

AMCL is an algorithm based on MCL but with adaptive parameters that help it adjust to changes in the environment over time. It maintains a probability distribution over the robot's pose. AMCL is commonly used in ROS-based navigation systems and is robust in environments with dynamic features.

Feature-Based Approaches:

Feature-based localization involves identifying and matching features (distinctive points or landmarks) in the environment with a map. Methods like Iterative Closest Point (ICP) or feature matching algorithms are often used. Feature-based approaches are suitable for scenarios where the environment has distinctive features that can be easily detected and matched.

3.5.5 Global Planner

The Global Planner is responsible for generating a high-level path from the robot's current position to the goal. It operates on a global costmap, which represents the overall structure of the environment. The primary goal is to find an optimal or near-optimal path that minimizes a predefined cost metric, considering the locations of obstacles and other constraints.

Different algorithms used for Global path planning:

Dijkstra's Algorithm:

Dijkstra's algorithm is a classic algorithm used for finding the shortest path between nodes in a graph. It explores nodes in order of their distance from the start node. Dijkstra's algorithm is suitable for scenarios where finding the globally optimal path is crucial, and the environment is relatively static.

A*(A-star):

A* is a widely-used pathfinding algorithm that guarantees the discovery of the shortest path in a weighted graph. It combines elements of both Dijkstra's algorithm and greedy best-first search. A* is suitable for static environments where the costmap is not expected to change frequently. It is effective in finding globally optimal paths.

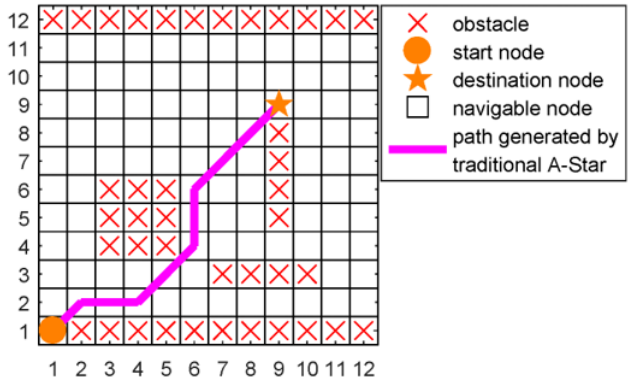


Figure 3.7: The grid map and path generated by the A-Star Liu *et al.* (2019)

Floyd-Warshall Algorithm:

The Floyd-Warshall algorithm is a dynamic programming approach used to find the shortest paths between all pairs of nodes in a weighted graph. Floyd-Warshall is more computationally expensive compared to A* and Dijkstra's algorithm but is suitable for scenarios where the global costmap is subject to frequent changes.

Out of the above 3 states algorithms, we use A-star algorithms at it relies on heuristic map updated and give a better estimate of path to the goal.

3.5.6 Local Planner

The Local Planner operates on a local costmap and focuses on adjusting the robot's trajectory to follow the global plan while handling dynamic obstacles and local conditions. It ensures that the robot adheres to the high-level plan while navigating through the immediate surroundings.

Algorithms:

Dynamic Window Approach (DWA):

DWA is a local planning algorithm that samples and evaluates potential robot velocities within a dynamically feasible window. It considers the robot's kinematics and dynamically adjusts its velocity based on the local costmap. DWA is suitable for real-time, dynamic environments and is capable of quickly adapting to changes in the robot's surroundings.

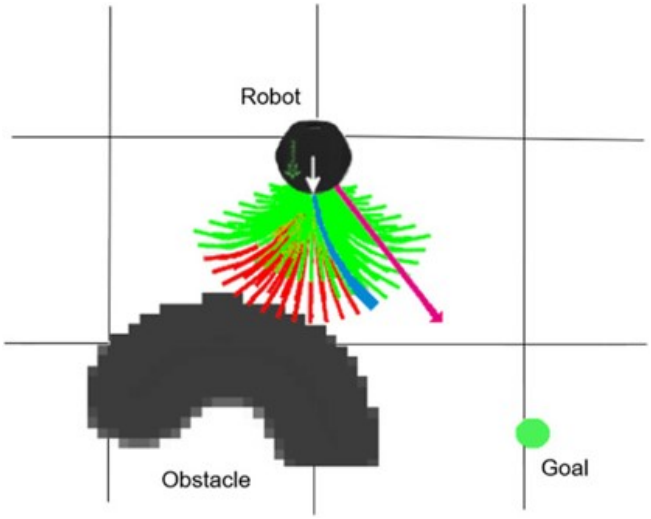


Figure 3.8: Admissible trajectories are shown by green color, whereas the red indicates the inadmissible trajectories. The magenda arrow and the blue show the heading angle and the best trajectory, respectively Hossain *et al.* (2022)

Trajectory Rollout:

Trajectory Rollout involves generating and evaluating multiple trajectories by simulating the robot's motion over a short time horizon. The trajectory with the least cost is selected for execution. Trajectory Rollout is effective in handling non-holonomic constraints and can adapt to both static and dynamic environments.

Teb Local Planner (Timed Elastic Band):

TEB is a local planner that represents the robot trajectory as a sequence of connected poses over time. It considers the robot's dynamic constraints and minimizes a cost function to generate feasible and collision-free trajectories. TEB is particularly useful for robots with non-holonomic constraints and is capable of handling dynamic obstacles effectively.

3.5.7 Behavior Tree and Task Execution

Navigation2 employs a Behavior Tree (BT) to coordinate the different tasks involved in navigation. The BT manages the sequencing and execution of actions such as planning, obstacle avoidance, and goal reaching. Each task is represented as a node in the BT, and the tree's structure dictates the order and conditions for task execution.

3.5.8 Recovery Behaviors

To handle unexpected situations or failures, Navigation2 incorporates recovery behaviors. These behaviors are triggered when the robot encounters difficulties, such as being stuck or facing an obstacle it cannot navigate around. Recovery behaviors aim to bring the robot back to a navigable state by, for example, attempting to rotate the robot or clearing the costmap.

3.5.9 ROS 2 Middleware

ROS 2 middleware facilitates communication between different components of the Nav2 stack. It allows the planner to receive sensor data, send commands to the robot's actuators, and exchange information with other modules seamlessly.

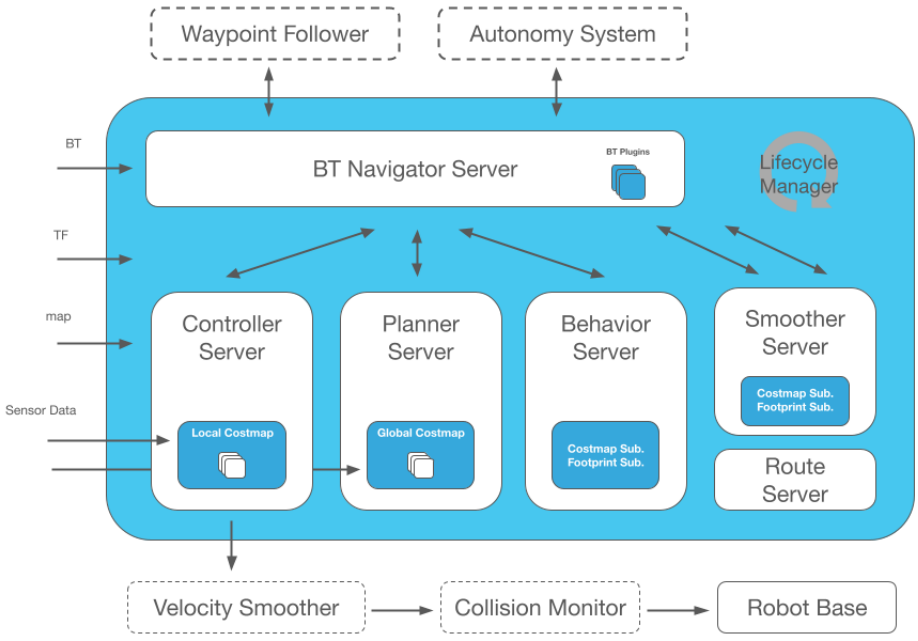


Figure 3.9: Navigation Architecture

3.5.10 Implementation of Inverse Kinematics Models

The 3-wheeled omnidirectional robot features a distinctive wheel configuration that grants it the ability to move in any direction with exceptional precision. Leveraging the advanced capabilities of the Navigation2 (Nav2) stack, we employ the principles of inverse kinematics (3.2.2) to seamlessly convert the desired body velocity, computed through Nav2's navigation algorithms, into the necessary angular velocities for each of the robot's wheels.

Our robot's kinematic structure comprises three omnidirectional wheels, positioned at 120-degree intervals relative to one another. Utilizing the body velocity information supplied by the Nav2 stack, we engage in a conversion process to translate it into individual wheel velocities. This conversion is facilitated by the application of inverse kinematics equations, tailored to our robot's specific geometry. These equations take into consideration the wheel angles, positions, and the overall geometry of the robot.

As the robot traverses its environment, the Nav2 stack continuously updates the desired body velocity in accordance with the global and local plans. The integration of the inverse kinematics module ensures that the wheel velocities are dynamically adjusted in real-time. This dynamic adjustment allows the robot to precisely adhere to the planned trajectory while adeptly navigating around obstacles in its path.

The incorporation of inverse kinematics into the Nav2 stack establishes a closed-loop control system. Within this system, the robot receives constant feedback on its actual motion, enabling it to make comparisons with the desired body velocity. Subsequently, the wheel velocities are adjusted accordingly, ensuring a high level of accuracy and responsiveness in the robot's navigation.

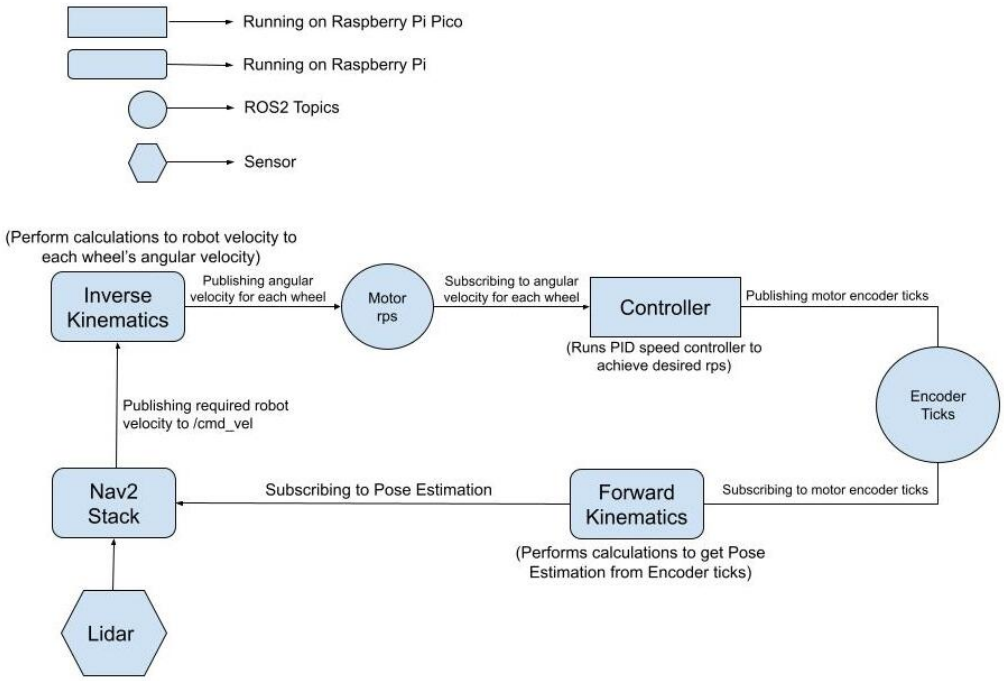


Figure 3.10: Controller Loop for Robot

3.5.11 Parameter Tuning Process

To ensure optimal performance of our robot's navigation system, extensive parameter tuning was conducted. The `param.yaml` file provided a centralized location for configuring various parameters governing the behavior of the Navigation2 (Nav2) stack. Below is an overview of the key parameters tuned and the rationale behind their adjustments:

Global Planner Parameters

- **Planner Type**: Initially, we experimented with both A* and Dijkstra planners to evaluate their performance in our environment. After extensive testing, we settled on the A* planner due to its ability to generate efficient paths while considering differential costs.
- **Timeouts**: The planner timeouts were adjusted based on the size of our environment and the complexity of the paths. Longer timeouts were set to allow the planner to explore larger search spaces without prematurely terminating planning tasks.

Local Planner Parameters

- **Planner Algorithm**: We evaluated both the Dynamic Window Approach (DWA) and Timed Elastic Band (TEB) algorithms for the local planner. TEB showed superior performance in dynamic environments with moving obstacles, so it was

selected for our robot.

- **Velocity Limits**: Maximum and minimum velocity limits were fine-tuned to ensure smooth and safe navigation, considering the physical constraints of our robot's motion capabilities.
- **Obstacle Avoidance Behavior**: Parameters controlling obstacle avoidance behavior, such as the cost function and obstacle inflation radius, were adjusted to achieve a balance between path clearance and proximity to obstacles.

Costmap Parameters

- **Inflation Radius**: The inflation radius of the costmap was carefully adjusted to account for the robot's size and clearance requirements. A larger radius was set to provide sufficient margin for maneuvering around obstacles while avoiding collisions.
- **Resolution**: We experimented with different costmap resolutions to balance computational efficiency with map accuracy. A finer resolution was chosen to capture detailed features of the environment without excessively increasing computational overhead.

Robot Footprint Parameters

- **Footprint Shape and Size**: The robot's footprint was accurately defined in the parameter file to reflect its physical dimensions. Fine adjustments were made to ensure precise collision checking and path planning.

Sensor Parameters

- **Laser Scanner Configuration**: Parameters governing the laser scanner, including scan range, angular resolution, and noise filtering, were optimized to enhance obstacle detection accuracy and reliability.

Localization Parameters

- **Localization Algorithm**: The Adaptive Monte Carlo Localization (AMCL) algorithm was selected for robust robot localization in our environment. Parameters such as particle filter count and update frequencies were tuned to achieve accurate pose estimation and minimize localization drift.

Follow Path Parameters

- Parameters such as minimum and maximum linear and angular velocities, along with acceleration, are defined. Additionally, goal tolerances are specified. Furthermore, important critics such as twirling, path alignment, and goal alignment are set.

By iteratively adjusting these parameters and conducting extensive testing in simulation and real-world scenarios, we achieved a navigation system tailored to the specific requirements and constraints of our robot and environment. The tuned parameters collectively contributed to improved navigation performance, robustness, and safety during operation.

3.5.12 Hardware Implementation

The implementation of hardware is began by selecting the materials and components that are going to be used in making the robot. Once the components are selected and before going to physically implement the project, the structure of the robot is designed in CAD using the dimensions of individual components and the planned dimensions for chassis. Based on the CAD design, the chassis is manufactured. Later, all the components are gradually checked if they are compatible and are working properly. In the process, few components had to be replaced by new or completely different components and few others had to be modified to make them compatible with other components. Then, all the components are assembled on to the chassis to make a final assembly (shown in Fig.) and attempts are made to set the robot into motion.

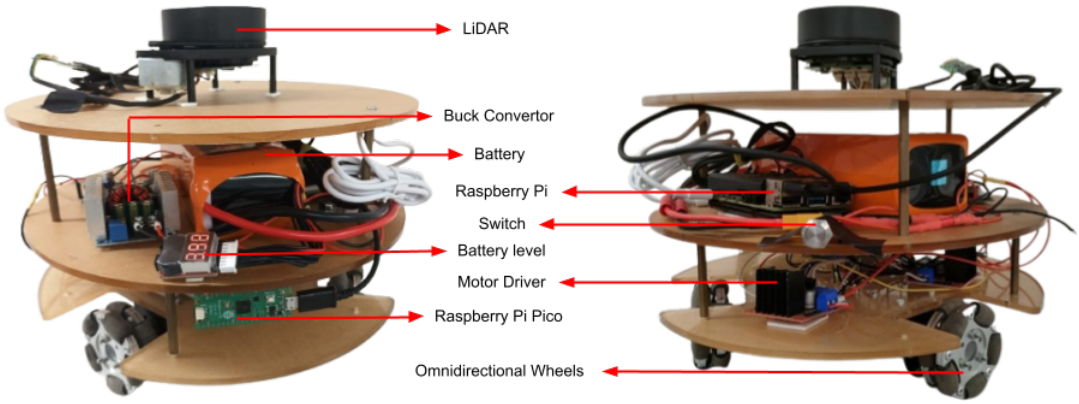


Figure 3.11: Robot Hardware

Component Selection:

Components are selected by balancing both the budget constraints and the physical requirements of the robot. A brief description on the selection of some important components is presented here.

- **Wheels:**
  Choosing the number and the type of wheels is crucial in building any wheeled

robotic system. Omni-wheeled robots can move in any direction without steering beforehand. Omni wheels have passive rollers with axes perpendicular to the wheels axis and these wheels slip when in motion. This enables the wheels to move holonomically, which means it can instantaneously move in any direction. Considering budget constraints and the feasibility of mathematical formulation for their motion, a three wheeled triangular configuration with omni wheels is chosen.

- **Motors:**
  DC Servo motors with specifications, 12V, 180 rpm, 3.5 kgcm, 7mm offset, and with builtin encoders are chosen to meet the speed, torque, and feedback requirements. Other factors like capacity of the power supply to drive the motors is also considered in selecting this motors.
- **Motor Drivers:**
  The L298N motor driver was chosen for its dual H-bridge configuration, enabling precise control of the 3-wheeled omnidirectional robot's motors. Its robust design and capability to handle the required current make it an ideal choice for our motor control needs.
- **Microprocessor:**
  The Raspberry Pi 4 with 4GB RAM and 32GB storage serves as the brain of our robotic system. Its powerful processing capabilities and extensive community support make it a reliable choice for running complex algorithms and managing the robot's overall control.
- **Microcontroller:**
  Raspberry Pi Pico board with RP2040 microcontroller, it is employed to manage low-level tasks and sensor interfacing. Its cost-effectiveness and ability to make direct communication with ROS2 environment makes it an ideal choice for distributed processing and sensor integration.
- **Battery:**
  The 24.2V 8000mAh LiPo Battery provides ample power for prolonged robot operation. Its rechargeable nature ensure extended runtimes, making it a suitable choice for powering the motors, electronics, and sensors while maintaining portability.
- **DC to DC Convertors:**
  They are employed to efficiently step down the voltage from the LiPo battery to both 12V for motors and 5V for raspberry pi. This ensures compatibility with various components, optimizing power distribution and enhancing the overall energy efficiency of the robotic system.
- **LiDAR:**
  The RPLidar A1-M8 lidar sensor was chosen for its 360-degree scanning capability and reliable obstacle detection. Its lightweight design and accurate ranging make it an essential component for robust SLAM and navigation applications.

3.6 Cooperative Navigation

Cooperative navigation, a sophisticated concept in robotics, entails the collaboration of multiple robotic entities in traversing intricate terrains with the aim of accomplishing a mutual navigational objective.

In these systems, robots share a plethora of data, including lidar scans, facilitating the construction of a comprehensive collective perception of the surroundings, essential for navigating through complex environments seamlessly.

Effective collaboration in cooperative navigation necessitates synchronized movements aimed at evading potential collisions while concurrently optimizing routes towards a common destination. Such collaborative endeavors rely on sophisticated distributed decision-making algorithms, strategically distributing tasks and resources among the robotic entities to ensure efficiency and efficacy.

One of the intriguing aspects of cooperative navigation systems is their capacity to exhibit emergent behavior, where the combined actions of multiple agents yield outcomes that surpass the capabilities of individual agents acting in isolation. This emergent behavior often leads to innovative solutions and adaptations to unforeseen challenges encountered during navigation.

The applications of cooperative navigation are diverse and impactful. They encompass a wide array of scenarios, including but not limited to search and rescue missions, where multiple robots can cover expansive areas in tandem, exploration missions in uncharted environments, where collaboration enhances mapping and understanding, and the coordination of autonomous vehicles within transportation networks, optimizing traffic flow and safety.

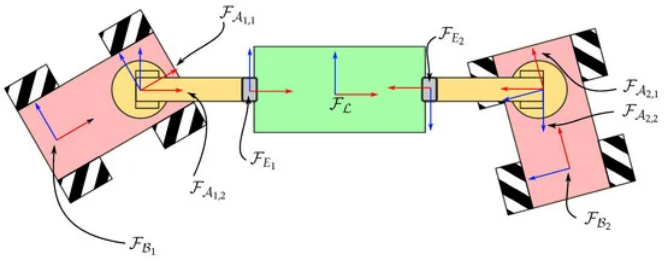


Figure 3.12: System consisting of 2 mobile manipulators carrying a rectangular object Vlantis *et al.* (2022)

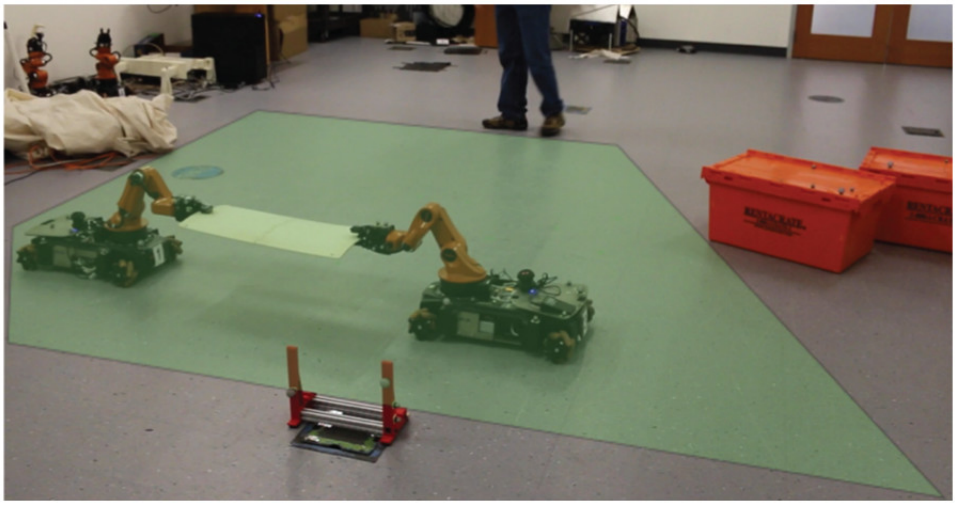


Figure 3.13: Two mobile manipulators collaboratively carry a rigid object. A projection of the obstacle-free convex region is superimposed in green. Alonso-Mora *et al.* (2017)

3.6.1 Cooperative Navigation Strategies

We experimented with various methodologies for cooperative navigation, employing configurations involving both two and three robots within the system. Among these methodologies, the primary approaches we explored include the Leader-Follower paradigm, the Distributed System model, and the Virtual Master approach.

Leader-Follower Approach

Goal is assigned to the center of the system robots which is translated to the leader robot and the other robots follow the leader at a specified distance and orientation.

Geometry of the system can be easily manipulated with this approach.

Based on Euclidean distance between each robot position and goal, robot with least distance is assigned as the leader.

Drawbacks –

1. Computational cost to run 3 navigation algorithm.
2. Geometric formation breaks down due to time lag in communication.
3. Minimum Euclidean distance was used to select dynamic leader, but that doesn't ensure shortest path.

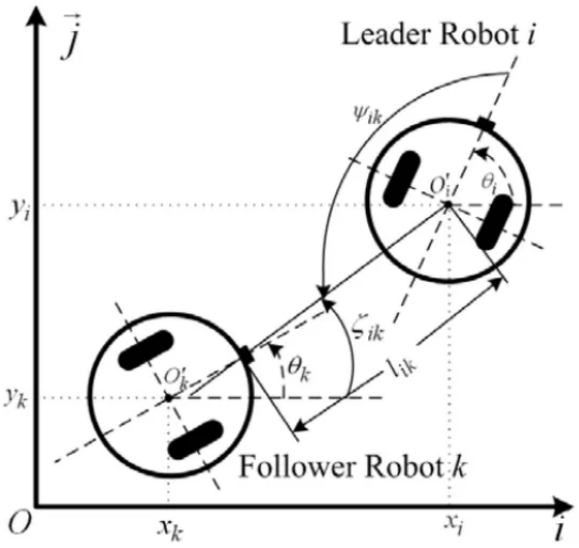


Figure 3.14: Leader-Follower schematic Qian and Xi (2018)

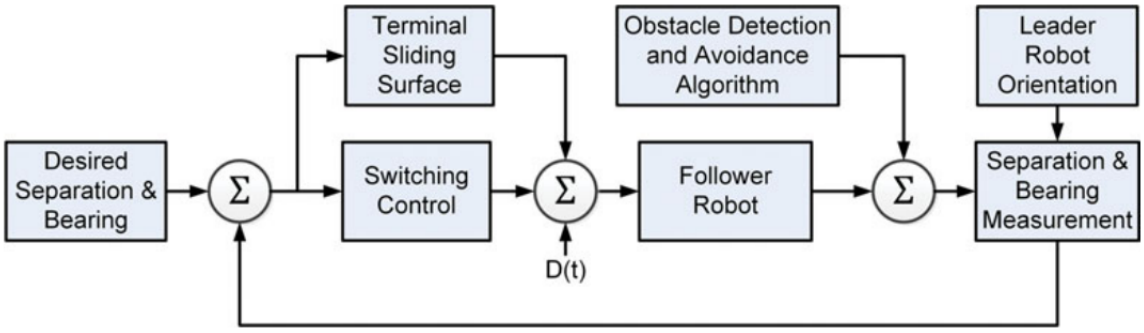


Figure 3.15: Leader-Follower approach Asif *et al.* (2017)

Distributed System Appraoch

This also follows similar logic as leader-follower but slightly different, here goal is assigned to the center of the system robots which is translated to all the 3 robots at a specified distance and orientation to have a triangular geometric formation throughout navigation.

Geometry of the system can be easily manipulated with this approach.

Based on Euclidean distance between each robot position and goal, robot with least distance is assigned as the leader.

Drawbacks –

1. Computational cost to run 3 navigation algorithm.
2. Geometric formation breaks down due to time lag in communication.

Virtual Master Approach

Centre of the system of the robots is assumed as a virtual master.

The individual robots in the system gets velocity from transforming velocity given to virtual centre at a specific distance orientation.

$$vel_{x\_left} = vel_x - \omega\dot{r} \tag{3.13}$$

$$vel_{y\_left} = vel_y \tag{3.14}$$

$$vel_{x\_right} = vel_x + \omega\dot{r} \tag{3.15}$$

$$vel_{y\_right} = vel_y \tag{3.16}$$

$$\omega_{left} = \omega_{right} = \omega \tag{3.17}$$

Similarly, the odometry data from each of the robot is used to calculate odometry for the virtual centre using following transformations,

$$x = \frac{x_{left} + x_{right}}{2} \tag{3.18}$$

$$y = \frac{y_{left} + y_{right}}{2} \tag{3.19}$$

$$\theta = \frac{\theta_{left} + \theta_{right}}{2} \tag{3.20}$$

$$vel_x = \frac{vel_{x\_left} + vel_{x\_right}}{2} \tag{3.21}$$

$$vel_y = \frac{vel_{y\_left} + vel_{y\_right}}{2} \tag{3.22}$$

$$\omega = \frac{\omega_{left} + \omega_{right}}{2} \tag{3.23}$$

where r, x, y, and θ are half of distance between center of robots, position of in x and y direction and orientation of the virtual centre. vel_x, vel_{x_left} and vel_{x_right} are x-direction velocities of virtual centre, left robot and right robot respectively. Similarly, vel_y, vel_{y_left} and vel_{y_right} are y-direction velocities of virtual centre, left robot and right robot respectively, and ω, ω_{left} and ω_{right} are y-direction velocities of virtual centre, left robot and right robot respectively

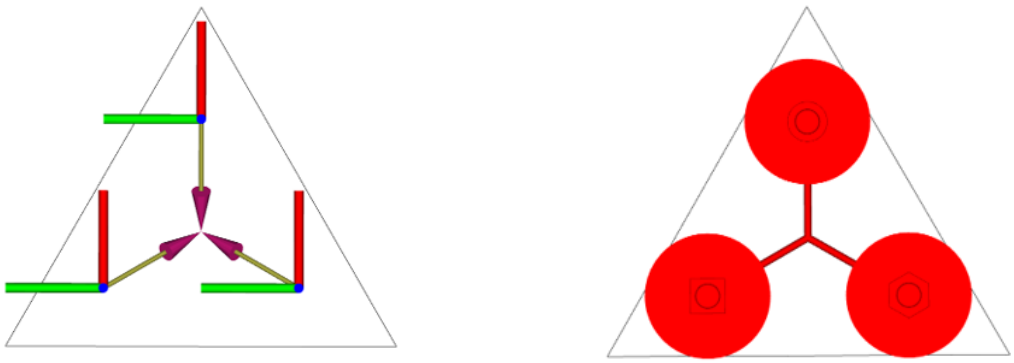


Figure 3.16: Triple Robot System with a virtual master

The system has a triangular footprint due to 3 robots. Apart from maintaining the geometric formation, the system is capable of changing its dimension. Computation cost is lower from having a single navigation algorithm operating.

3.6.2 Algorithm for Changing Dimension of Robot System

The map obtained from SLAM consists of two files, namely .pgm and .yaml. The .pgm file is a greyscale image file with 3 colours: white(255) pixels representing free space, black(0) pixels representing obstacles and grey representing unknown regions. .yaml files contained the map's details such as origin coordinate, threshold values for occupied pixels and free pixels, and map resolution, i.e., meters per pixel.

Several stages of image processing is executed to obtain the obstacle coordinates from the .pgm file, this involved rotation, erosion dilation thresholding, corner detection and origin translation of the image origin to origin in the environment.

Two-Dimensional Obstacles

In the designed environment, the obstacles are represented in two dimensions, characterized by their length and height. To determine the distance between obstacles, the endpoints of each obstacle are detected. By analyzing these endpoints, the system calculates the distance between the obstacles.

Next, the trajectory of the robot is taken into account. Line segments are formed between the detected corner points of the obstacles within a specified threshold distance.

If a line segment intersects with the robot's trajectory and the length of the line segment is less than the width of the robot's base footprint, a reconfiguration of the robot's dimensions is triggered, and navigation continues. This adjustment ensures that the robot can pass through the gap between the obstacles without colliding with them.

However, if the length of the line segment exceeds the width of the robot's base footprint, no reconfiguration is necessary, and the robot continues navigation without modification to its dimensions.

This adaptive approach to obstacle detection and navigation allows the robot to dynamically adjust its dimensions when needed, ensuring safe and efficient traversal through the environment while minimizing disruptions to its path.

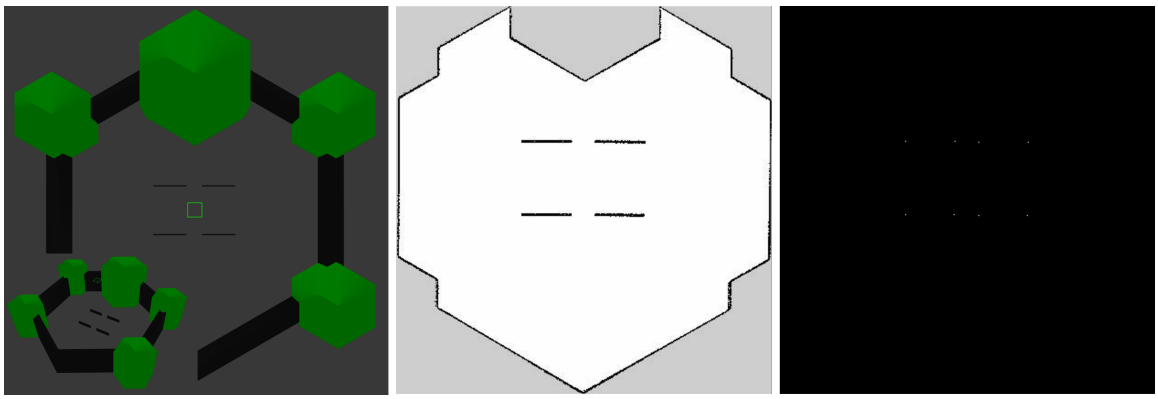


Figure 3.17: Image Processing for 2D Obstacles

Three-Dimensional Obstacles

In the context of 3D obstacles, the determination of edge coordinate pixels is achieved through Canny edge detection. To initiate obstacle detection, a point along the robot's trajectory is selected, typically at a predefined distance from the robot's center. From this point, a perpendicular line to the trajectory is drawn, extending outwards.

This perpendicular line is then analyzed to ascertain if it intersects with any obstacles within the environment. Should intersections occur, the distances between the points of intersection are calculated. If this distance is found to be less than the dimensions of the robot's footprint, it signifies that the robot cannot navigate through the gap without potential collision.

In such a scenario, navigation is momentarily paused, and the dimensions of the robot's footprint are dynamically adjusted to accommodate the passage through the

gap. However, if the calculated distance between the intersection points exceeds the dimensions of the robot's footprint, navigation proceeds without any changes to the robot's configuration.

This method enables the robot to adapt its dimensions based on real-time obstacle detection, ensuring safe traversal through the environment while minimizing the risk of collisions with obstacles. By dynamically modifying its footprint when necessary, the robot can navigate through narrow passages and complex environments effectively.

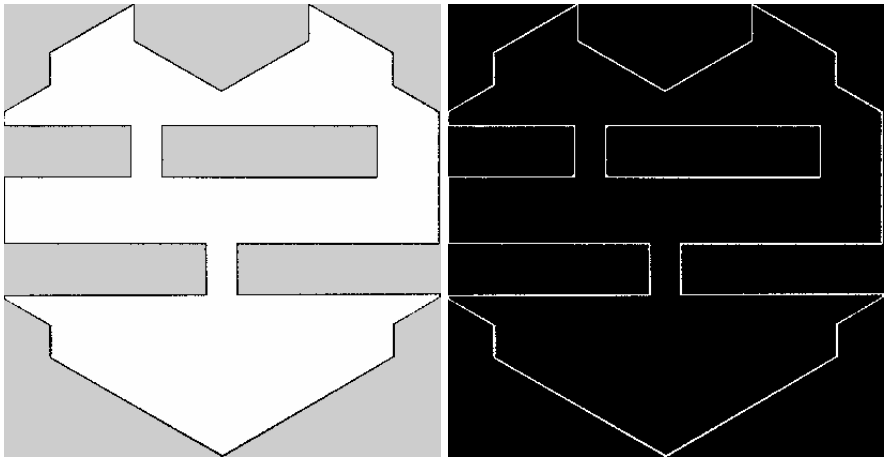


Figure 3.18: Image Processing for 3D Obstacles

3.6.3 Dynamic Footprint Algorithm

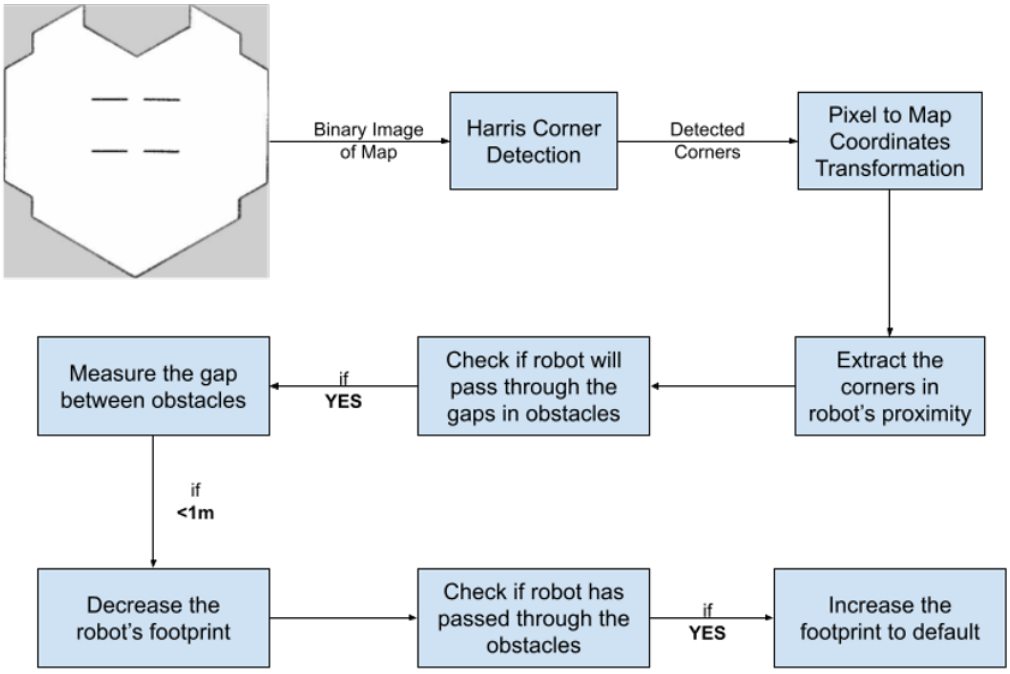


Figure 3.19: Flowchart of Dynamic Footprint Algorithm

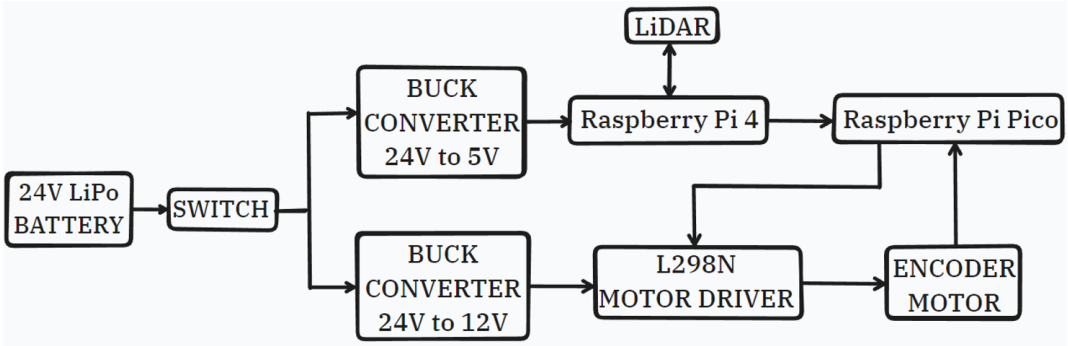
In a ROS 2 environment, when modifications are made to the system configuration and dimensions of a robot, typically encoded in an XACRO (XML Macros) file, the updated file needs to be uploaded. This action involves replacing the previous configuration with the new one. Once uploaded, the revised XACRO file is published to the '*/robot_description*' topic.

This process is crucial in the ROS (Robot Operating System) navigation stack, as it ensures that the navigation system accurately interprets the robot's physical characteristics and environment. By updating the configuration and dimensions, the robot's behavior in navigation tasks can be refined and optimized according to its current specifications.

The '*/robot_description*' topic serves as a centralized communication channel within the ROS ecosystem, allowing components such as the navigation stack to access the most up-to-date description of the robot. This ensures consistency and coherence between various modules and facilitates seamless integration of new configurations into the navigation pipeline.

Overall, the process of updating the robot configuration via XACRO files and publishing them to the '*/robot_description*' topic is fundamental for maintaining accurate and efficient robot navigation in ROS 2 environments.

3.7 System Architecture



3.8 Challenges

Having started from scratch, we had faced many difficulties and challenges throughout the project. The major challenges faced in designing a system that processes, controls the motion, and guides the robots to perform their tasks are associated with path planning. The performance of the functions used in autonomous navigation is sensitive to the changes on its various parameters and values. So finding suitable parameters had been a challenge. Apart from the challenges presented above, major challenges were faced while working on hardware, especially with electronics. The hubs for joining the wheels to the motor shaft were not delivered as per specifications and turned out to be incompatible. There were issues in setting up and configuring motor drivers and a microcontroller board. These orders were non-returnable and the customer support was very poor. These challenges helped in realising the complexity and uncertainty involved in engineering practice and some of the possible ways to tackle them.

Another challenge encountered during the project revolved around the tuning of PID (Proportional, Integral, Derivative) values for the speed controller in the robot. Achieving optimal performance required a delicate balance between these parameters to ensure stable and accurate movements. The process involved iterative adjustments, considering factors such as wheel slippage and the robot's responsiveness. Fine-tuning the PID values was essential to mitigate issues like overshooting or sluggish responses, contributing to the overall stability and efficiency of the robot's navigation system.

In cooperative navigation scenarios, where multiple robots operate simultaneously within the same environment, several challenges arise, notably, running multiple navigation stacks concurrently can impose a significant computational burden on the system, especially in resource-constrained environments or when dealing with complex maps and algorithms. This increased computational demand may lead to delays in decision-making, decreased overall system responsiveness, or even system failures. Also, when multiple robots traverse the same or interconnected paths, the cumulative effect of slip or localization errors can compound, leading to inaccuracies in estimating each robot's pose. This is particularly problematic in scenarios where precise coordination or alignment between robots is necessary, such as collaborative manipulation tasks or formation control.

CHAPTER 4

Results and Discussions

4.1 Single Robot

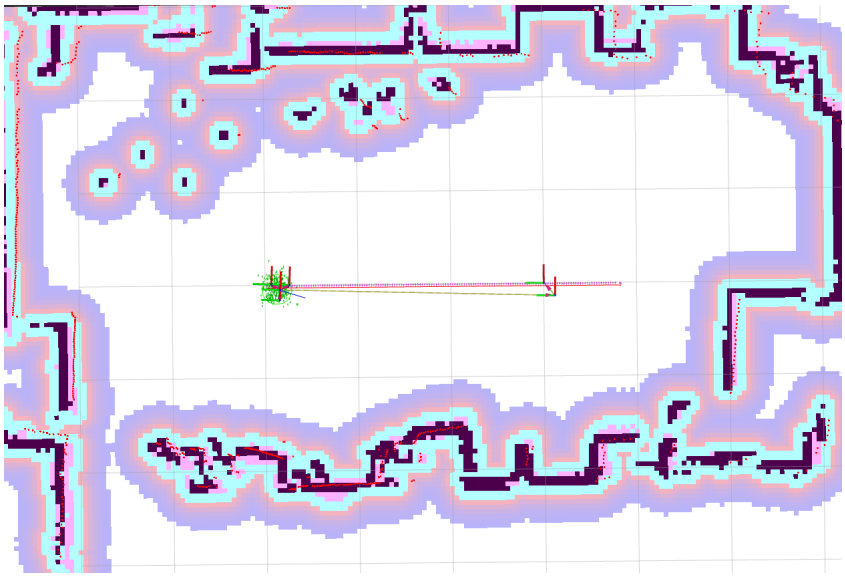


Figure 4.1: Path Planning by single robot

The Fig.(4.1) illustrates a clearly defined blue line, representing the robot's planned path. This trajectory is indicative of the system's capability to generate an optimal route towards the predefined goal, effectively navigating through the environment and circumventing obstacles.

Surrounding the robot are adaptive Monte Carlo localization (AMCL) particle arrows. These arrows dynamically convey the system's estimated pose, providing real-time feedback on the localization confidence. This feature contributes to the robustness of the robot's spatial awareness during navigation.

The laser scan data, a fundamental component of the navigation stack, is visualized in the RViz2 window. This data enables the robot to create an accurate occupancy grid, facilitating obstacle avoidance and real-time decision-making. The responsiveness of the

laser scan data is evident in the robot's ability to navigate through complex environments with precision.

Two distinct maps, the local map and the global map, are discernible in the Fig.(4.1). The local map (more vibrant in colour) represents the immediate surroundings of the robot, updating in real-time based on laser scan data. In contrast, the global map offers a comprehensive view of the entire environment, aiding the robot in long-term planning and goal-oriented navigation.

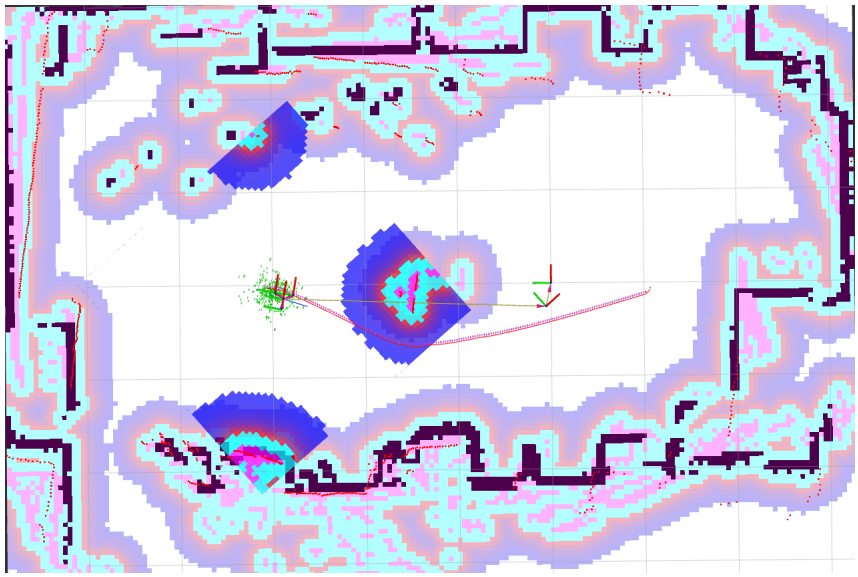


Figure 4.2: Dynamic Path Planning by single robot

In the Fig.(4.3), the robot encounters a dynamic obstacle disrupting its initially planned path. Here's where it gets interesting. The system, equipped with real-time perception and decision-making capabilities, swiftly responds to the obstacle and recalculates a new trajectory.

The blue planned path dynamically adjusts to circumvent the newly introduced obstacle. The Fig.(4.3) provides a visual representation of the dynamic obstacle (in vibrant colors), allowing us to observe how the robot perceives and reacts to changes in its surroundings. The laser scan data, in conjunction with obstacle detection algorithms, enables the robot to identify the dynamic obstacle and promptly alter its course.

The robot smoothly transitions from the original path to the revised trajectory. The robot gracefully navigates around the dynamic obstacle, avoiding any potential collision. This capacity for dynamic obstacle avoidance enhances the overall safety and efficiency of the robot's navigation in dynamic environments.

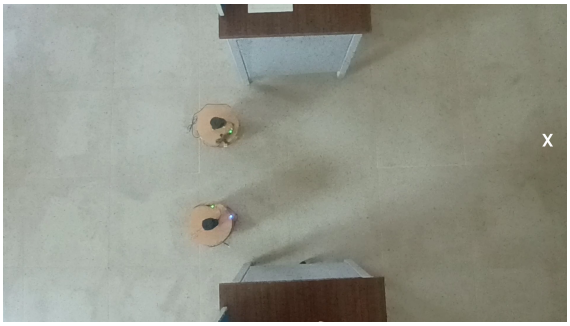
The AMCL particle arrows play a crucial role in this adaptive behavior. As the robot navigates around the dynamic obstacle, observe how the particle arrows adjust, indicating the robot's evolving perception of its own position. This adaptability contributes to maintaining accurate localization even in the presence of unforeseen obstacles.

4.2 Multi-Robot System

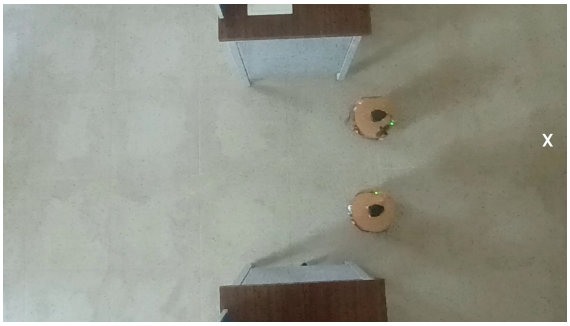
In the depicted scenario of dual robot configuration, the system operates in a default state where the centers of the robots are positioned at a fixed distance of 1 meter apart, as illustrated in Figure 4.3. This default configuration is maintained as long as no obstacles are detected within the environment.

Figure 4.3: Default Configuration of Dual Robot System

However, when an obstacle is detected, as depicted in Figure 4.4a, the system initiates a dynamic reconfiguration process. Upon calculating the distance between the obstacles and verifying that it is less than the dimensions of the robots in their current configuration, the system determines that a gap exists that the robots can traverse. Consequently, the distance between the robots is reduced, and the configuration is adjusted to fit through the gap, as illustrated in Figure 4.4b.



(a) Reducing footprint of Dual Robot System



(b) Increasing footprint of Dual Robot System

Figure 4.4: Dynamic footprint variation of Dual Robot System

Once the multi-robot system successfully navigates through the gap, as shown in Figure 4.4b, it returns to its default configuration, where the robots are positioned at the original distance of 1 meter apart, as depicted in Figure 4.5.

This dynamic reconfiguration mechanism enables the multi-robot system to adapt to changes in the environment, navigate through narrow passages, and maintain efficient coordination while minimizing the risk of collisions with obstacles. By autonomously adjusting their configuration based on real-time sensor data and environmental conditions, the robots can effectively navigate complex environments and achieve their objectives with enhanced flexibility and safety.

Figure 4.5: Goal achieved by Dual Robot System

CHAPTER 5

SUMMARY AND CONCLUSION

Throughout this project, we have established the groundwork for a cooperative system of 3-wheel omnidirectional robots, each equipped with an advanced laser sensor, operating seamlessly under the guidance of Simultaneous Localization and Mapping (SLAM). Commencing from the conceptual stage, we developed a sophisticated program encompassing AMCL for localization, path planning utilizing algorithms like A-star, and optimization of generated paths for precise navigation.

The robot showcases impeccable performance within simulated environments. The implemented algorithms effectively execute processes, demonstrating the success of our efforts in achieving accuracy. The successful implementation of inverse and forward kinematic models has been pivotal in ensuring the robot's precision and adaptability.

While the program thrives within controlled settings, the scope for future refinement is extensive, both in terms of augmenting the control program and advancing the hardware implementation. Real-world applications necessitate additional considerations, such as adapting pre-processing techniques for images and precise calibration of parameters tailored for our robotic platform.

Potential avenues for future enhancements include the development of an adaptive actuation system, enabling precise positioning of objects at the geometric center within various configurations formed by multiple robots. This adaptive capability would empower the robots to dynamically reconfigure themselves, steering clear of obstacles while executing tasks.

Successfully implemented a cooperative robot system with multiple robots working collaboratively. Through meticulous design and fine-tuning, optimized mobility, sensor integration, and navigation capabilities of our initial robot, laying a solid foundation for the architecture and functionality of the cooperative system.

REFERENCES

1. **J. Alonso-Mora**, **S. Baker**, and **D. Rus** (2017). Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, **36**(9), 1000–1021. URL https://doi.org/10.1177/0278364917719333.

2. **M. Asif**, **M. J. Khan**, and **A. Y. Memon** (2017). Integral terminal sliding mode formation control of non-holonomic robots using leader follower approach. *Robotica*, **35**(7), 1473–1487.

3. **C. Debeunne** and **D. Vivet** (2020). A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, **20**(7). ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/20/7/2068.

4. **W. P. N. dos Reis**, **O. Morandin**, and **K. C. T. Vivaldini**, A quantitative study of tuning ros adaptive monte carlo localization parameters and their effect on an agv localization. *In 2019 19th International Conference on Advanced Robotics (ICAR)*. 2019.

5. **H. Durrant-Whyte** and **T. Bailey** (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, **13**(2), 99–110.

6. **T. Hossain**, **H. Habibullah**, **R. Islam**, and **R. V. Padilla** (2022). Local path planning for autonomous mobile robots by integrating modified dynamic-window approach and improved follow the gap method. *Journal of Field Robotics*, **39**(4), 371–386. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22055.

7. **K. Y. K. Leung**, **T. D. Barfoot**, and **H. H. T. Liu**, Decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks. *In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010.

8. **C. Liu**, **Q. Mao**, **X. Chu**, and **S. Xie** (2019). An improved a-star algorithm considering water current, traffic separation and berthing for vessel path planning. *Applied Sciences*, **9**(6). ISSN 2076-3417. URL https://www.mdpi.com/2076-3417/9/6/1057.

9. **D. Nuss**, **S. Reuter**, **M. Thom**, **T. Yuan**, **G. Krehl**, **M. Maile**, **A. Gern**, and **K. Dietmayer** (2016). A random finite set approach for dynamic occupancy grid maps with real-time application.

10. **J. Ota** (2006). Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics*, **20**(1), 59–70. ISSN 1474-0346. URL https://www.sciencedirect.com/science/article/pii/S1474034605000509.

11. **D. Qian** and **Y. Xi** (2018). Leader–follower formation maneuvers for multi-robot systems via derivative and integral terminal sliding mode. *Applied Sciences*, **8**(7). ISSN 2076-3417. URL https://www.mdpi.com/2076-3417/8/7/1045.

12. **R. Siegwart**, **I. R. Nourbakhsh**, and **D. Scaramuzza**, *Mobile Robot Kinematics*. 2011, 57–99.

13. **T. Tao**, **Y. Huang**, **J. Yuan**, **F. Sun**, and **X. Wu**, Cooperative simultaneous localization and mapping for multi-robot: Approach experimental validation. *In 2010 8th World Congress on Intelligent Control and Automation*. 2010.

14. **P. Vlantis**, **C. P. Bechlioulis**, and **K. J. Kyriakopoulos** (2022). Mutli-robot cooperative object transportation with guaranteed safety and convergence in planar obstacle cluttered workspaces via configuration space decomposition. *Robotics*, **11**(6). ISSN 2218-6581. URL https://www.mdpi.com/2218-6581/11/6/148.

15. **B. Zhang**, **J. Liu**, and **H. Chen**, Amcl based map fusion for multi-robot slam with heterogenous sensors. *In 2013 IEEE International Conference on Information and Automation (ICIA)*. 2013.

APPENDIX A

Technical Specifications of Components used

Parts	**Specifications**	**Quantity**
Microprocessor Board	Raspberry Pi 4 B with 4GB RAM and 32GB storage running on Ubuntu 22.04 server	2
Microcontroller Board	Raspberry Pi Pico with Dual-core Arm Cortex-M0+ @ 133MHz, 264KB on-chip SRAM and 2MB on-board QSPI flash	2
Motor Drivers	L298N Dual H Bridge DC Motor Driver	4
Omni wheels	60mm Diameter	6
Battery	22.2V 8000mAh 25C 6S Lithium Polymer Battery	1
Battery	11.1V 10000mAh 25C 3S Lithium Polymer Battery	1
DC-to-DC converter	300W 20A DC-DC Buck Converter Step-down Module	4
LiDAR	Range upto 6m and Sample Duration: 0.5 ms	2
Terminal Block	0.08-2.5mm 6:2 Pole Wire Connector	2
Motors	180 rpm and torque of 3.5 kg-cm with encoder feedback	6
Acrylic Discs	Laser cut 270mm diameter and 5 mm thick	6

Table A.1: Technical Specification of Components