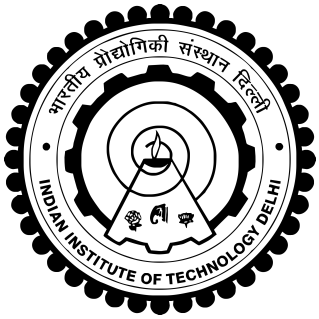
# RESEARCH INTERNSHIP

INTERNSHIP REPORT


## ADITHYA R

*Under the guidance of*
**PROF. S K SAHA**
*Indian Institute of Technology Delhi*

*Internship Mentor*
**MR. SRIKRISHNA S.**
*Director, Seianmai Technologies*



**INDIAN INSTITUTE OF TECHNOLOGY DELHI**
*HAUZ KHAS, NEW DELHI-110016 INDIA*


**NOVEMBER 2023 - JULY 2024**

# Indian Institute of Technology Delhi
## Department of Mechanical Engineering
Hauz Khas, New Delhi 110016, INDIA
Tel: (011)2659 1135,
Email: saha@mech.iitd.ac.in
http://sksaha.com

**Dr. S.K. Saha**
Professor


July 12, 2024

## Internship Certificate

This is to certify that **Adithya R** worked at **SeiAnmai Technologies** (Incubated with **IHFC, IIT Delhi**) mentored by **Srikrishna S** and under the supervision of **Prof. S.K. Saha** from **01/11/2023** to **15/07/2024** as an Intern. During this period, He worked on enhancing the functionality of the TOTO robot, a mobile telepresence and tele-observance robot.

Adithya R contributed to the development of anti-crash logic, parameter tuning of the NAV2 stack, autonomous mapping using a wave frontier-based algorithm, and the integration of an assistance system for seamless user interaction. These projects significantly improved the robot's safety, efficiency, and interactivity.

Adithya R demonstrated an outstanding ability to apply the knowledge learned through college coursework and self-interest in robotics. The practical insights and significant learning opportunities gained during this internship have been invaluable. He also prepared comprehensive documentation and submitted a detailed report of the work accomplished.

Prof . S. K Saha,
Dept. of Mechanical Engineering,
IIT Delhi

# ACKNOWLEDGMENTS

I would like to express my profound gratitude to Prof. S K Saha, who provided me with the opportunity to undertake this research internship under his guidance. His invaluable support, insightful suggestions, and encouragement have been instrumental in the successful completion of my research. I am immensely grateful to Mr. Srikrishna S., Director at Seianmai Technologies, for his mentorship from the beginning of my internship. His practical knowledge, constant guidance, and experience have greatly contributed to my learning and growth. His ability to solve complex problems and his leadership skills have been inspiring and motivational throughout my internship. I would also like to extend my appreciation to the entire team at Seianmai Technologies for creating a conducive and collaborative work environment. The shared knowledge and team spirit have been truly enriching. Special thanks to my family and friends for their unwavering support and encouragement, which have been a constant source of motivation. Their belief in my abilities has been a significant driving force in my academic and professional journey. Finally, I would like to acknowledge the Indian Institute of Technology Delhi for providing me with the platform to undertake this research internship and for the resources and support that made this work possible. The exposure to cutting-edge research and technology has been invaluable.

Adithya R

# ABSTRACT

This report presents the work conducted during my research internship at Seianmai Technologies, under the guidance of Prof. S K Saha from the Indian Institute of Technology Delhi. The internship focused on enhancing the functionality of the TOTO robot, a mobile telepresence and tele-observance robot. Key contributions include the development of anti-crash logic, parameter tuning for smoother navigation, autonomous mapping using a wave frontier-based algorithm, and an assistance system integrating ChatGPT for voice interaction.

The anti-crash logic ensures safe navigation by identifying obstacles and adjusting the robot's speed accordingly. Parameter tuning of the NAV2 stack enhanced the robot's movement efficiency, making it smoother and more reliable. The autonomous mapping algorithm, integrated with SLAM, allows the robot to create detailed maps of its environment. Additionally, the assistance system leverages Porcupine for wake word detection and Whisper.cpp for transcription, enabling seamless interaction with users through ChatGPT.

This report details the methodologies, implementations, and results of these contributions. The experiences gained from this internship have been invaluable, providing practical insights and significant learning opportunities in the field of robotics.

**Keywords:** Mobile Robots, Autonomous Navigation, Telepresence, Tele-observance, SLAM, Anti-crash Logic, Parameter Tuning, Voice Interaction, ChatGPT.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **AMCL** | Adaptive Monte Carlo Localization |
| **BMS** | Battery Management System |
| **CMU** | Carnegie Mellon University |
| **DOA** | Direction of Arrival |
| **DDS** | Data Distribution Service |
| **HD** | High Definition |
| **LIDAR** | Light Detection and Ranging |
| **NLP** | Natural Language Processing |
| **PID** | Proportional Integral Derivative |
| **ROS2** | Robot Operating System 2 |
| **RViz** | ROS Visualization Tool |
| **SLAM** | Simultaneous Localization and Mapping |
| **TTS** | Text-to-Speech |
| **VAD** | Voice Activity Detection |
| **WWD** | Wake Word Detection |

# Chapter 1
# INTRODUCTION

## 1.1 About Seianmai Technologies

Seianmai Technologies is a technology company that offers Internet of Robotics solutions for home, office, industry, and education. The company provides a complete package that includes a customized interface, an open-source software stack, and customizable hardware designed for mobile robots. Currently, the company is developing TOTO Robot, which stands for Tele-Operation and Tele-Observance Robot.[1]


Figure 1.1: Logo of Seianmai Technologies

### 1.1.1 Vision and Mission

Seianmai Technologies aims to begin with the educational robot market and expand to telepresence robots. The long-term vision includes providing a

comprehensive home/office robotics solution to improve productivity within the next five years. The mission of Seianmai Technologies is to be a leading provider of robotic solutions that integrate seamlessly into various operational environments, enhancing efficiency and productivity through advanced robotics technology.

### 1.1.2 Core Competencies

The company specializes in several key areas of robotics technology, including:

- **SLAM (Simultaneous Localization and Mapping):** Utilizing industry-standard ROS and ROS2 frameworks to enable robots to create detailed maps of their environment.
- **Navigation:** Developing advanced navigation algorithms to ensure efficient and reliable movement of robots in dynamic environments.

## 1.2 TOTO Robot

TOTO Robot is used for telepresence, meaning making someone's presence felt at a remote location, and teleobservance, meaning observing and seeing another location area without being physically present. The two versions of TOTO Robot currently in development are the TOTO Commercial Edition Robot (TOTOCE) and the TOTO Research Edition Robot (TOTORE). The

main functionalities of TOTO Robot include live streaming of video through the robot's camera, manual and autonomous operation in dynamic environments, efficient navigation and localization in unknown places, obstacle detection and avoidance, and a customizable, easy-to-use user interface.

### 1.2.1 TOTO Robot Commercial Edition

TOTO Robot Commercial Edition is designed for direct use in homes, offices, and factories. TOTOCE features an advanced, compact, and reliable design suitable for any environment. It also has an arm-type configuration and an attached tablet to interact with the robot.

Figure 1.2: TOTO Commercial Edition Robot

### 1.2.2 TOTO Robot Research Edition

TOTO Robot Research Edition is specially designed for researchers, educational purposes, students, and other institutions for educating robotics. TOTORE features numerous customization options, open-source software drivers, and rapid testing functionality. It operates on the leading industry-oriented ROS2 framework and includes a LIDAR sensor and a camera for autonomous navigation, Simultaneous Localization and Mapping (SLAM), and an automatic docking system. The TOTORE robot uses a Raspberry Pi 4 as the main computer for processing.

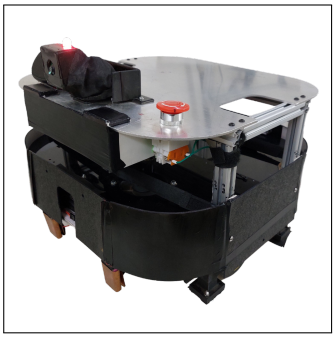


Figure 1.3: TOTO Research Edition Robot

## 1.3 Tasks Accomplished During Internship

During the internship, various tasks were undertaken to enhance the functionality and performance of the TOTO robot. These tasks included extensive

research, rapid prototyping, and multiple testing phases of various hardware and software algorithms.

### 1.3.1 Anti-crash Logic Development

The development of anti-crash logic was a crucial task to ensure the safe navigation of the TOTO robot while controlling it manually. This logic involves identifying obstacles in the robot's path and adjusting its speed accordingly. The robot scans the area in front of it using LIDAR sensors and takes data from +30 degrees to -30 degrees relative to its current heading. If an obstacle is detected within this range, the robot's speed is gradually reduced. If the obstacle is less than 0.3 meters away, the robot comes to a complete stop to avoid a collision. This feature significantly enhances the robot's ability to navigate in dynamic environments safely.

### 1.3.2 Parameter Tuning

Parameter tuning of the NAV2 stack was performed to enhance the robot's movement efficiency. This task involved adjusting various parameters within the navigation stack to ensure smoother and more reliable navigation. The tuning process included optimizing the robot's acceleration, deceleration, and turning parameters to achieve a balance between speed and stability. By fine-tuning these parameters, the robot can navigate more effectively, avoiding obstacles and reaching its goals with greater precision.

### 1.3.3 Autonomous Mapping

Autonomous mapping was achieved by integrating a wave frontier-based algorithm with SLAM (Simultaneous Localization and Mapping). This task enabled the TOTO robot to create detailed maps of its environment autonomously. The wave frontier-based algorithm improves the efficiency of the mapping process by prioritizing unexplored areas and systematically expanding the known map. This integration allows the robot to navigate unknown environments, continuously updating its map and improving its localization accuracy.

### 1.3.4 Assistance System Integration

An assistance system was developed leveraging Porcupine for wake word detection and Whisper.cpp for transcription. This system enables seamless interaction with users through ChatGPT. When the robot hears the wake word, it activates its voice recognition system and starts transcribing the user's commands using Whisper.cpp. The transcribed text is then sent to ChatGPT, which processes the command and generates an appropriate response. This response can include actions for the robot to perform or information to relay back to the user. This integration enhances the robot's capability to interact with users naturally and effectively.

# Chapter 2

# ANTI-CRASH LOGIC

## 2.1 Task Overview

The implementation of the anti-crash logic in the TOTO robot aims to enhance operational safety by dynamically adjusting the robot's trajectory and speed in response to detected obstacles, using data primarily from LIDAR sensors. The objective is to prevent collisions in a wide range of operational environments, from tightly controlled lab settings to dynamic, real-world scenarios.

This feature was developed to address the potential risk of the robot crashing when manually driven by a user. Although the robot had a feature called assistive teleoperation to help avoid obstacles, it relied on the robot's costmap, which increased computational load. Given access to the robot's internals, we sought to improve this logic by developing a more efficient anti-crash system.

## 2.2 Approach and Implementation

The approach to implementing the anti-crash logic involved understanding the robot's movement dynamics and how it interacts with obstacles. This section explains the theory behind the working of the algorithm.

### 2.2.1 Derivation of Movement and Detection Angles

Let's consider the robot as a 2D planar, where the obstacle is at coordinates $(x, y)$, and the robot is moving toward the obstacle with a constant linear velocity $V_x$ and angular velocity $\omega_z$. We aim to find the angle $\theta$, which is the angle of the obstacle from the robot.

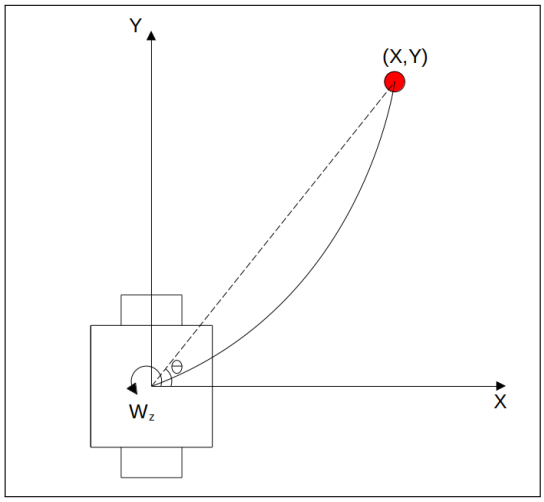



Figure 2.1: Robot at initial position

Given the relationship $\tan\theta = \frac{y}{x}$, let's consider $t = 1$ s to calculate the robot's position after 1 second of travel.

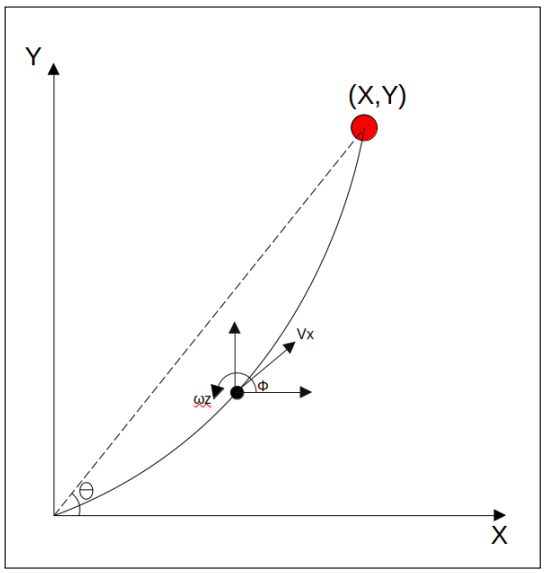



Figure 2.2: Robot position after 1 second of travel

Since $V_x$ and $\omega_z$ are constant, we have the following differential equations:

$$\frac{dx}{dt} = V_x \cos(\phi) \tag{2.1}$$

$$\frac{dy}{dt} = V_x \sin(\phi) \tag{2.2}$$

where $\phi$ is the angle the robot makes with the x-axis. The rate of change of $\phi$ is given by $\omega_z$:

$$\frac{d\phi}{dt} = \omega_z \tag{2.3}$$

Integrating this equation over time $t$:

$$\phi = \int_0^t \omega_z dt = \omega_z t \tag{2.4}$$

Substituting $\phi$ into the equations for $x$ and $y$:

$$x = \int_0^t V_x \cos(\omega_z t) dt = \frac{V_x}{\omega_z} \sin(\omega_z t) \tag{2.5}$$

$$y = \int_0^t V_x \sin(\omega_z t) dt = \frac{V_x}{\omega_z} (1 - \cos(\omega_z t)) \tag{2.6}$$

Substituting these into the expression for $\tan\theta$:

$$\theta = \tan^{-1} \left( \csc(\omega_z t) - \cot(\omega_z t) \right) \tag{2.7}$$

Thus, the angle to be scanned is identified. The range of the scan is $\theta - \theta' \leq \theta \leq \theta + \theta'$, where $\theta'$ is the additional angle to be scanned for safety.

### 2.2.2 Speed Adjustment and Stopping Mechanism

The robot must slow down gradually when approaching an obstacle. Let $d$ be the minimum laser scan distance between the robot and the obstacle. If $d > 1$ meter, the robot's speed $V_x$ is adjusted according to the function $f(d)$:

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} - f(d) \tag{2.8}$$

When the obstacle is within 0.3 meters, the robot must stop:

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} - f(0.3) = 0 \tag{2.9}$$

We need to find $f(d)$ that satisfies these conditions. Assuming a linear expression for $f(d)$:

$$f(d) = md + c \tag{2.10}$$

We substitute the conditions into this linear expression:

$$m(0.3) + c = V_{x_{\text{in}}} \tag{2.11}$$

$$m(1) + c = 0 \tag{2.12}$$

Solving for $m$ and $c$:

$$m = -c \tag{2.13}$$

$$0.3m - m = V_{x_{\text{in}}} \tag{2.14}$$

$$-0.7m = V_{x_{\text{in}}} \implies m = -\frac{V_{x_{\text{in}}}}{0.7} \tag{2.15}$$

Substituting $m$ into $f(d)$:

$$f(d) = \left(-\frac{V_{x_{\text{in}}}}{0.7}\right) d + \frac{V_{x_{\text{in}}}}{0.7} \tag{2.16}$$

Simplifying:

$$f(d) = \frac{V_{x_{\text{in}}}}{0.7}(1 - d) \tag{2.17}$$

Substituting $f(d)$ into the expression for $V_{x_{\text{out}}}$:

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} - f(d) \tag{2.18}$$

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} - \frac{V_{x_{\text{in}}}}{0.7}(1 - d) \tag{2.19}$$

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} \left(1 - \frac{1 - d}{0.7}\right) \tag{2.20}$$

$$V_{x_{\text{out}}} = V_{x_{\text{in}}} \left(\frac{d - 0.3}{0.7}\right) \tag{2.21}$$

Thus, the required linear velocity is calculated such that the robot will gradually reduce speed as it approaches the obstacle, stopping completely when the obstacle is 0.3 meters away.

## 2.3 Algorithm Overview

The anti-crash logic algorithm is designed to dynamically adjust the robot's speed based on the distance to detected obstacles, ensuring safe navigation. The algorithm uses data from the robot's LIDAR sensors to identify obstacles and compute the necessary adjustments to the robot's velocity.

**Step 1: LIDAR Data Collection** The robot continuously collects data from its LIDAR sensors to detect obstacles in its path. The LIDAR sensors provide a 360-degree view of the surroundings, allowing the robot to identify obstacles in all directions.

**Step 2: Angle Calculation** Based on the detected obstacles, the algorithm calculates the angle $\theta$ between the robot's current trajectory and the obstacles. This angle helps in determining the direction and proximity of the obstacles.

**Step 3: Speed Adjustment** The robot's speed is adjusted according to the distance to the obstacles. If the distance $d$ is greater than 1 meter, the speed $V_x$ is reduced gradually as the robot approaches the obstacle. If the distance is less than 0.3 meters, the robot stops completely to avoid collision.

**Step 4: Continuous Monitoring** The algorithm continuously monitors the robot's surroundings and adjusts the speed in real-time based on the updated LIDAR data. This ensures that the robot can navigate safely in dynamic environments.

This approach to anti-crash logic ensures that the robot can operate safely

and efficiently, avoiding collisions and navigating through various environments with ease. The use of real-time data from LIDAR sensors allows for immediate response to potential obstacles, enhancing the overall safety and reliability of the robot.

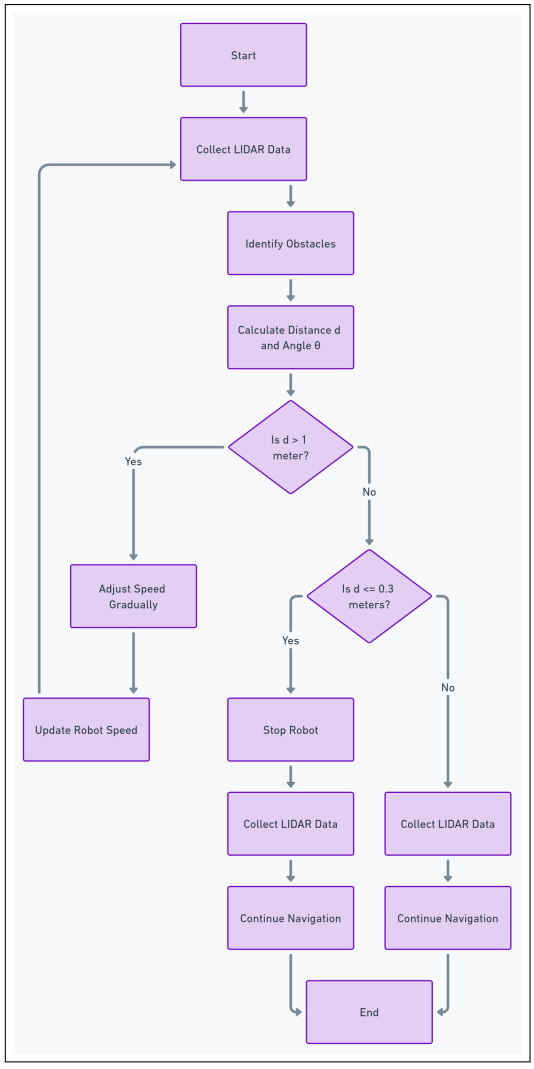



Figure 2.3: Flowchart of Anti-Crash Logic

# Chapter 3

# PARAMETER TUNING

## 3.1 Task Overview

Initially, the TOTO robot faced challenges in reaching its goals and avoiding obstacles effectively. The robot would not stop even when it seemed to reach its goal, and it often collided with dynamic obstacles that appeared in its path. The path planning was not optimal, leading to inefficient navigation. To address these issues, extensive parameter tuning was conducted to improve the robot's navigation, obstacle avoidance, and path planning capabilities. This chapter discusses the modifications made to various parameters in the robot's navigation stack, resulting in more robust and smooth navigation performance.

## 3.2 Approach and Implementation

The tuning of parameters involved adjusting several components within the robot's navigation stack, including AMCL (Adaptive Monte Carlo Localization), behavior tree navigator, controller server, and costmap configurations. Each parameter was meticulously adjusted and tested in real-time to observe its effects on the robot's navigation.[2]

### 3.2.1 Overview of NAV2

NAV2 (Navigation 2) is a ROS2 package designed to provide autonomous navigation capabilities to robots. It includes several components that work together to allow a robot to move from a starting point to a goal location while avoiding obstacles. Key components of NAV2 include:

- **AMCL (Adaptive Monte Carlo Localization):** A localization system that uses a particle filter to track the position and orientation of the robot within a known map.
- **Behavior Tree Navigator:** A framework that uses behavior trees to define and execute navigation tasks, allowing for more flexible and modular navigation strategies.
- **Controller Server:** Responsible for executing control algorithms that guide the robot along a path while avoiding obstacles.

- **Costmaps:** Used to represent the environment around the robot, including static and dynamic obstacles, to facilitate path planning and obstacle avoidance.

### 3.2.2 AMCL (Adaptive Monte Carlo Localization)

The AMCL algorithm was fine-tuned by adjusting the noise model parameters to better reflect the robot's motion characteristics and environment. Key parameters adjusted include:

- **alpha1, alpha2, alpha3, alpha4, alpha5:** These parameters define the noise in the motion model. Increasing these values makes the algorithm more tolerant to noise but less responsive to actual motion.
  - **alpha1:** Controls linear noise in translational motion.
  - **alpha2:** Similar to alpha1, controls linear noise in translational motion.
  - **alpha3:** Controls rotational noise.
  - **alpha4:** Controls rotational noise, particularly useful in handling sudden orientation changes.
  - **alpha5:** Controls linear noise in rotational motion.
- **beam_skip_distance, beam_skip_error_threshold, beam_skip_threshold:** These parameters relate to beam skipping in the laser model, balancing between computational efficiency and localization accuracy.

  - **beam_skip_distance:** Set to 0.5 meters, defining the minimum distance for beam skipping.
  - **beam_skip_error_threshold:** Set to 0.9, determining the error threshold for beam skipping.
  - **beam_skip_threshold:** Set to 0.3, defining the acceptable fraction of beams to skip.

- **max_particles, min_particles:** Defines the number of particles in the particle filter.
  - **max_particles:** Set to 500 for a more accurate representation of the robot's belief.
  - **min_particles:** Set to 100 to ensure diversity in the particle set.

### 3.2.3 Behavior Tree Navigator

The behavior tree navigator parameters were tuned to optimize the robot's path planning and goal-reaching efficiency:

- **bt_loop_duration:** Set to 10 seconds, defining the main loop duration.
- **default_server_timeout:** Set to 20 seconds, specifying the timeout for server calls.
- **controller_frequency:** Set to 20 Hz, indicating how often the controller server updates and sends control commands.

- **progress_checker_plugin:** Set to "progress_checker", monitoring the robot's progress towards the goal.
- **goal_checker_plugins:** Includes "general_goal_checker" to evaluate goal achievement.
- **controller_plugins:** Includes "FollowPath", guiding the robot along predefined paths.

### 3.2.4 Controller Server

The controller server parameters were fine-tuned to enhance the robot's movement precision and responsiveness:

- **min_x_velocity_threshold:** Set to 0.02, defining the minimum linear velocity in the x-axis.
- **min_y_velocity_threshold:** Set to 0.0, indicating no minimum threshold for y-axis velocity.
- **min_theta_velocity_threshold:** Set to 0.05, defining the minimum angular velocity.
- **failure_tolerance:** Set to 0.3, specifying the deviation tolerance before triggering a failure response.
- **acc_lim_x, acc_lim_theta:** Set to 1.0, indicating the maximum acceleration limits in linear and angular directions.

- **decel_lim_x, decel_lim_theta:** Set to -1.0, defining the maximum deceleration limits.

### 3.2.5 Costmap Configuration

The costmap parameters were adjusted to improve obstacle detection and avoidance:

- **local_costmap:**
  - **update_frequency:** Set to 5.0 Hz for more frequent updates.
  - **publish_frequency:** Set to 5.0 Hz for regular updates to other components.
  - **robot_radius:** Set to 0.25 meters, influencing obstacle avoidance margins.
  - **plugins:** Includes "obstacle_layer" and "inflation_layer" for detailed costmap information.
- **global_costmap:**
  - **update_frequency:** Set to 1.0 Hz, balancing between responsiveness and computational load.
  - **publish_frequency:** Set to 1.0 Hz, ensuring regular updates.
  - **robot_radius:** Set to 0.25 meters, maintaining consistency with the local costmap.

### 3.2.6 Visualization with RViz

RViz is a 3D visualization tool for ROS that allows users to see what the robot perceives and plans in real-time. It was extensively used during the parameter tuning process to visualize the following:

- **Robot Model:** Displaying the robot's model, including its sensors and moving parts.
- **Laser Scan Data:** Visualizing the LIDAR data to see obstacles detected by the robot.
- **Costmaps:** Showing both global and local costmaps to understand the environment representation and how obstacles are marked.
- **Planned Path:** Visualizing the path planned by the navigation stack, helping in understanding and debugging the robot's movement.
- **Localization:** Displaying the robot's estimated position and orientation, helping in fine-tuning the AMCL parameters.

### 3.2.7 Simulation with Gazebo

Gazebo is a powerful simulation tool that integrates with ROS to provide a realistic environment for testing robots. It was used to simulate the TOTO robot's performance in various scenarios:

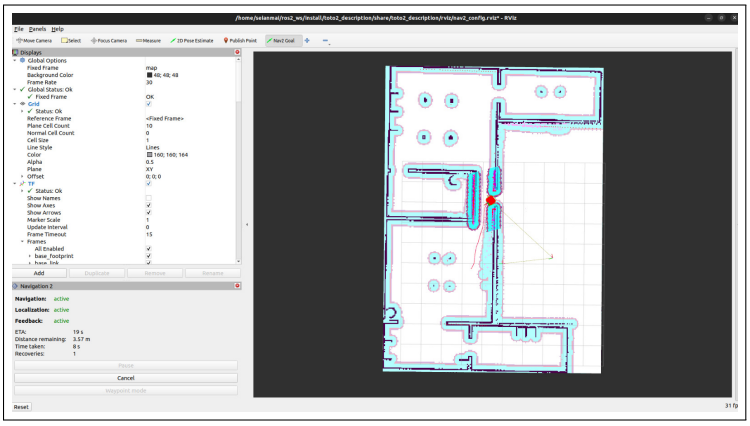


Figure 3.1: RViz Visualization of Robot's Navigation

- **Environment Simulation:** Creating different environments, from simple rooms to complex mazes, to test navigation and obstacle avoidance.

- **Sensor Simulation:** Simulating the LIDAR, cameras, and other sensors to test perception algorithms.

- **Real-Time Testing:** Running real-time simulations to observe the robot's behavior and make necessary parameter adjustments without risking hardware damage.

- **Performance Metrics:** Collecting data on navigation performance, such as time to goal, path efficiency, and obstacle avoidance success rate.

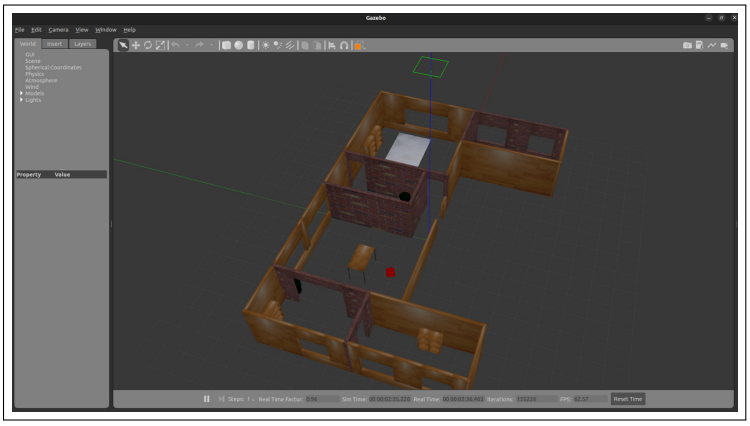


Figure 3.2: Gazebo Simulation of Robot's Environment

## 3.3 Results and Testing

The tuning of these parameters significantly improved the TOTO robot's navigation and obstacle avoidance performance. The robot now demonstrates more robust and smooth movement, effectively reaching its goals while avoiding obstacles. Key outcomes include:

### 3.3.1 Improved Navigation Performance

The adjustments in AMCL and behavior tree navigator parameters resulted in more accurate localization and efficient path planning. The robot could navigate complex environments with fewer deviations from the planned path.

### 3.3.2 Enhanced Obstacle Avoidance

Tuning the controller server and costmap configurations improved the robot's ability to detect and avoid obstacles dynamically. The robot could adjust its speed and trajectory in real-time, ensuring safe operation.

### 3.3.3 Issues Faced and Solutions Implemented

During the tuning process, several issues were encountered, including:

- **Localization Drift:** Adjusting the AMCL noise parameters helped mitigate localization drift, ensuring the robot maintained accurate positioning.
- **Path Deviation:** Fine-tuning the behavior tree navigator and controller server parameters reduced path deviation, enabling the robot to follow its planned trajectory more closely.
- **Obstacle Detection Latency:** Increasing the update frequencies for local and global costmaps improved obstacle detection responsiveness, reducing latency and enhancing safety.

Overall, the parameter tuning process has significantly enhanced the TOTO robot's operational capabilities, making it more reliable and efficient in various environments.

# Chapter 4

# AUTONOMOUS MAPPING BASED ON WAVE FRONTIER ALGORITHM

## 4.1 Task Overview

The implementation of autonomous mapping using the wave frontier algorithm in the TOTO robot aims to enhance the robot's capability to explore unknown environments and create detailed maps autonomously. This functionality is essential for various applications, including telepresence and teleobservance, where the robot must navigate and understand its surroundings independently. Initially, when providing the robot to the user, there is a chance that the user may map the area incorrectly or be unfamiliar with the mapping process. The autonomous mapping feature simplifies this process for the user.

## 4.2 Approach and Implementation

To achieve this, I researched various algorithms for autonomous mapping techniques and found the wave frontier-based algorithm to be highly effective. The wave frontier algorithm was implemented in the TOTO robot to facilitate autonomous exploration and mapping. This section details the approach and implementation, including the integration with the ROS2 framework, the utilization of simulation environments, and the steps involved in the algorithm.[7]

### 4.2.1 Wave Frontier Algorithm

The wave frontier algorithm involves the following key steps:

- **Initialization:** The ROS2 node is initialized with the name "nav2_waypoint_tester". The node is derived from 'rclcpp::Node'.
- **Subscriptions:**
  - Subscribes to the "/odom" topic to receive odometry data.
  - Subscribes to the "/map" topic to receive occupancy grid data.
- **Action Client:**
  - Utilizes the 'NavigateToPose' action client to send navigation goals.
- **Costmap Management:**

  - Creates a costmap from the received occupancy grid data.
  - Identifies free space and frontier points based on costmap data.

- **Frontier Detection and Navigation:**
  - Detects frontier points and calculates centroids for navigation goals.
  - Navigates to the identified frontiers to explore the environment.

The following sections explain the key components and their roles in the algorithm.

### 4.2.2 Initialization and Subscriptions

The ROS2 node is initialized with the name "nav2_waypoint_tester", and it subscribes to the necessary topics:

- `"/odom"`: Receives odometry data, which provides the robot's current position and orientation.
- `"/map"`: Receives the occupancy grid data, which is used to create a costmap for navigation.

### 4.2.3 Action Client

The node uses the 'NavigateToPose' action client to send navigation goals to the robot. This action client enables the robot to navigate to specified poses by sending goals that the robot will attempt to reach.

### 4.2.4 Costmap Management

The costmap is created from the received occupancy grid data. This costmap is used to identify free space, obstacles, and unknown areas. The costmap is essential for detecting frontiers and planning navigation paths.

### 4.2.5 Frontier Detection and Navigation

The core of the wave frontier algorithm is the detection of frontiers and navigating to them. Frontiers are boundaries between known and unknown areas in the map. The algorithm follows these steps:

1. Convert the robot's current pose from world coordinates to map coordinates.

2. Identify free space in the costmap using a breadth-first search (BFS).

3. Detect frontier points and classify them based on their proximity to free space and obstacles.

4. Calculate the centroid of each frontier and use it as a navigation goal.

5. Send the navigation goals to the robot using the 'NavigateToPose' action client.

6. The robot iteratively moves to new frontiers until no more frontiers are detected.

### 4.2.6 Working of the Wave Frontier Algorithm

The working of the wave frontier algorithm involves the following steps:

1. **Initialization:** The ROS2 node is initialized, and subscriptions to the necessary topics are set up.

2. **Receive Data:** The robot receives odometry data from the '/odom' topic and occupancy grid data from the '/map' topic.

3. **Create Costmap:** A costmap is created from the occupancy grid data, identifying free space, obstacles, and unknown areas.

4. **Convert Pose:** The robot's current pose is converted from world coordinates to map coordinates.

5. **Identify Free Space:** Free space is identified in the costmap using a breadth-first search (BFS).

6. **Detect Frontiers:** Frontier points are detected and classified based on their proximity to free space and obstacles.

7. **Calculate Centroids:** The centroid of each frontier is calculated and used as a navigation goal.

8. **Send Navigation Goals:** Navigation goals are sent to the robot using the 'NavigateToPose' action client.

9. **Move to Frontiers:** The robot iteratively moves to new frontiers until no more frontiers are detected.

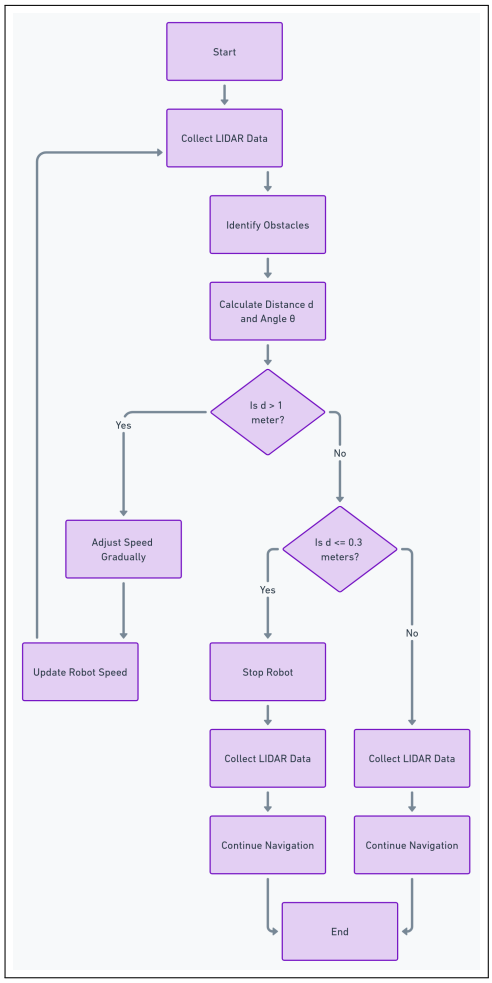



Figure 4.1: Flowchart of the Wave Frontier Algorithm

### 4.2.7 Simulation in Gazebo and RViz

The initial implementation and testing of the wave frontier algorithm were conducted in the Gazebo and RViz simulation environments. These tools provided a controlled setting to develop, visualize, and refine the algorithm before deploying it on the actual robot.

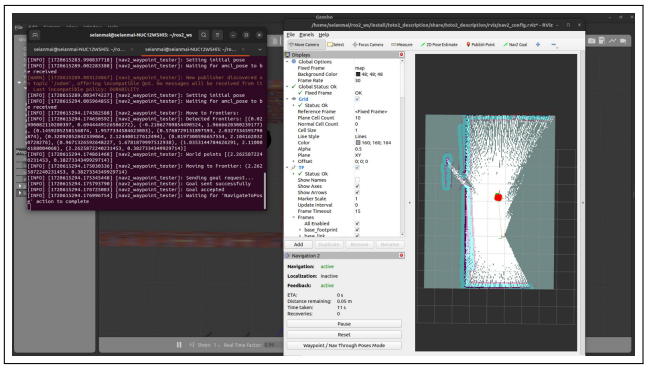


Figure 4.2: Initial scan data in the simulation environment. The robot starts with an initial position and scans the environment to identify free space and obstacles.

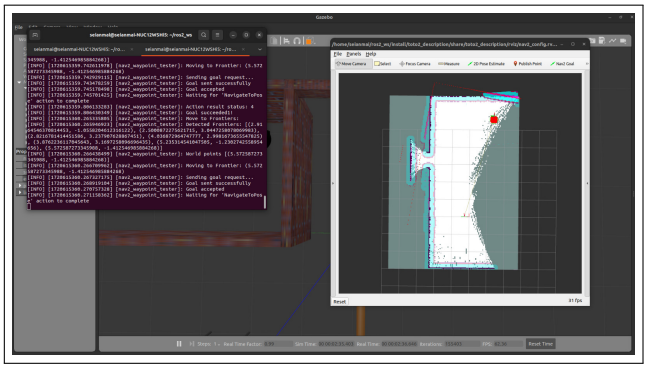


Figure 4.3: Robot exploring detected frontiers in simulation. The robot moves towards the frontiers to explore unknown areas and update the map.

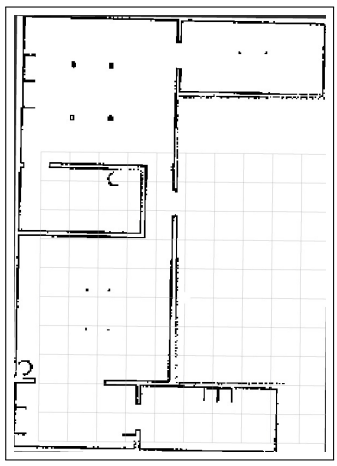


Figure 4.4: Final map generated in the simulation environment. The robot successfully explores the environment and creates a detailed map.

### 4.2.8 Deployment and Testing on Actual Robot

After successful testing in simulation, the wave frontier algorithm was deployed on the TOTO robot. During real-world testing, the robot was able to explore and map most of the environment successfully. However, certain issues were encountered, such as small gaps in the real environment creating frontiers, which required manual intervention to stop the program once the area was sufficiently mapped. Additionally, running navigation and SLAM simultaneously with this algorithm demanded significant computational power, which impacted performance.

## 4.3 Results and Testing

The autonomous mapping functionality was tested extensively in both simulated and real environments. The following outcomes and observations were recorded:

### 4.3.1 Simulation Results

In the Gazebo and RViz simulations, the robot was able to explore and map the entire environment without manual intervention. The frontier detection and navigation were smooth, and the final maps were accurate and detailed.

### 4.3.2 Real-World Testing

In real-world tests, the robot successfully explored and mapped the majority of the environment. However, small gaps and obstacles sometimes created frontiers that required manual stopping of the program. Despite these minor issues, the overall performance was satisfactory, with the robot demonstrating robust and efficient navigation and mapping capabilities.

## 4.4 Conclusion

The implementation of the wave frontier algorithm in the TOTO robot significantly enhanced its autonomous mapping capabilities. The detailed simulation tests in Gazebo and RViz ensured a robust implementation before

real-world deployment. The real-world tests validated the algorithm's effectiveness, with only minor issues that were easily manageable. However, the computational demands and small gap frontiers pose challenges that require further optimization. This program currently requires improvements and optimization to be used effectively in real-world applications, considering the computational constraints. At present, due to these constraints, the autonomous mapping feature is not being actively used in the robot, but the foundation laid by this implementation provides a promising direction for future developments.

# Chapter 5

# Voice Interaction System

## 5.1 Task Overview

The implementation of the voice interaction system in the TOTO robot aims to enable the robot to respond to voice commands accurately and efficiently. The system involves several key components, including Voice Activity Detection (VAD), Wake Word Detection (WWD), Speech Transcription, Command Processing using ChatGPT, and Text-to-Speech (TTS) response generation. This section provides an overview of the tasks accomplished, the challenges faced, and the solutions implemented.

## 5.2 Approach and Implementation

The approach to implementing the voice interaction system involved several critical steps, starting from understanding the requirements and researching suitable technologies to integrating the components into a cohesive system.

Each of these steps is detailed below.

### 5.2.1 Voice Activity Detection (VAD)

Voice Activity Detection (VAD) is a technique used to detect the presence of human speech in audio signals. It is crucial for distinguishing between speech and non-speech segments, allowing the system to process only the relevant parts of the audio, thereby improving efficiency and accuracy.

The initial approach involved researching various VAD techniques to identify a reliable method. Several techniques were tested, including Sphinx[10][12], which required Python 2, making it incompatible with the ROS2 framework used in the project. The WebRTC-VAD [8][9]library was also explored, but building the library presented significant challenges.

Eventually, the Whisper library was selected for VAD due to its robustness and compatibility with ROS2. The library provided accurate voice activity detection, and its integration with the system allowed for real-time processing of audio data.

### 5.2.2 Wake Word Detection (WWD)

Wake Word Detection (WWD) is the process of identifying specific predefined keywords or phrases (wake words) in an audio stream. The detection of these wake words triggers the system to start processing the subsequent speech. This is essential for activating the system only when needed, conserving

computational resources and power.

For wake word detection, the Porcupine library[13] was chosen due to its high accuracy and low computational requirements. The library was integrated into the ROS2 framework, enabling the robot to detect predefined wake words such as "hey TOTO". Upon detecting the wake word, the system would trigger the transcription process, allowing the robot to process and respond to voice commands.

### 5.2.3 Speech Transcription

Speech Transcription is the process of converting spoken language into written text. It involves capturing audio data, processing it to recognize speech, and generating the corresponding text output. This is a critical step in voice interaction systems as it translates spoken commands into a format that can be processed by the system.

The Whisper library[15]was used for speech transcription. The library's support for C++ made it a suitable choice for the project, allowing for efficient processing of audio data. The transcription process involved capturing audio data, converting it to text, and publishing the transcribed text to the appropriate ROS2 topics for further processing.

The initial implementation faced challenges with high CPU usage, which were mitigated by optimizing the library to focus on predefined commands rather than general voice recognition. This approach reduced the computational load and improved the system's responsiveness.

### 5.2.4 Command Processing and Response Generation

Command Processing involves interpreting the transcribed text to determine the appropriate actions the robot should take. This is achieved using advanced natural language processing (NLP) techniques. In this project, the ChatGPT model from OpenAI was used for command processing.

ChatGPT[16] is a state-of-the-art language model capable of understanding and generating human-like text based on the input it receives. For this project, the transcribed text was processed using the ChatGPT API, which generated appropriate responses based on the commands received.

### 5.2.5 Text-to-Speech (TTS)

Text-to-Speech (TTS)[17] is the process of converting written text into spoken words. This allows the robot to provide verbal responses to user commands, enhancing interaction and communication.

The OpenAI TTS model was used to convert the generated responses into speech. The TTS functionality provided by OpenAI ensured high-quality and natural-sounding speech, making interactions with the robot more intuitive and effective.

### 5.2.6 Direction of Arrival (DOA) and Robot Movement

The Direction of Arrival (DOA) system is used to determine the direction from which the voice command is coming. This information is crucial for orienting the robot towards the speaker, improving its responsiveness and interaction capabilities.

The DOA and Robot Movement node receives direction information from the DOA system. Based on the direction information, the node calculates the target angle and adjusts the robot's orientation to face the sound source. Once the robot is oriented correctly, it starts listening for further commands.

## 5.3 System Components and Integration

The voice interaction system for the TOTO robot consists of several integrated components. This section provides a detailed explanation of each component and its role in the system.

### 5.3.1 Voice Activity Detection (VAD)

Voice Activity Detection (VAD) identifies segments of audio that contain speech, distinguishing them from background noise and silence. This step is crucial for processing only relevant audio data, improving the efficiency and accuracy of subsequent steps. The Whisper library was used for VAD due to

its robustness and compatibility with ROS2.

### 5.3.2 Wake Word Detection (WWD)

Wake Word Detection (WWD) involves identifying specific predefined keywords or phrases in the audio stream, such as "hey TOTO". Upon detecting the wake word, the system activates and starts processing the subsequent speech. The Porcupine library was chosen for WWD due to its high accuracy and low computational requirements.

### 5.3.3 Speech Transcription

Speech Transcription converts spoken language into written text, which can then be processed by the system. The Whisper library was used for this purpose, leveraging its support for C++ to ensure efficient processing of audio data. The transcription process captures audio, converts it to text, and publishes the transcribed text to ROS2 topics for further processing.

### 5.3.4 Command Processing with ChatGPT

Command Processing interprets the transcribed text to determine the appropriate actions for the robot. This is achieved using the ChatGPT model from OpenAI, which is a state-of-the-art language model capable of understanding and generating human-like text based on input. The ChatGPT node processes the transcribed text and generates appropriate responses.

### 5.3.5 Text-to-Speech (TTS)

Text-to-Speech (TTS) converts written text into spoken words, allowing the robot to provide verbal responses to user commands. The OpenAI TTS model was used to generate high-quality, natural-sounding speech, enhancing the robot's interaction capabilities.

### 5.3.6 Direction of Arrival (DOA) and Robot Movement

The Direction of Arrival (DOA) system determines the direction from which a voice command is coming, enabling the robot to orient itself towards the speaker. The DOA and Robot Movement node calculates the target angle based on direction information and adjusts the robot's orientation accordingly. Once oriented, the robot starts listening for further commands.

### 5.3.7 Integration and Workflow

The integration of these components forms a cohesive workflow for the voice interaction system. The process begins with the Direction of Arrival (DOA) system, which is always active, detecting the direction of sound. When the wake word detection (WWD) system identifies the wake word (e.g., "hey TOTO"), it sends the DOA data to the robot, which then turns to face the direction of the sound.

Once the robot completes turning, it starts listening for further commands. The speech is then transcribed using the Whisper library, and the transcribed text is sent to the ChatGPT node for processing. ChatGPT generates a response and identifies the appropriate action based on the command.[18]

The workflow is as follows:

1. **DOA Detection:** The DOA system continuously monitors the direction of sound.

2. **Wake Word Detection:** Upon detecting the wake word, the WWD system sends the DOA data to the robot.

3. **Robot Orientation:** The robot turns to face the direction of the sound.

4. **Speech Transcription:** After turning, the robot starts listening and transcribing the speech using the Whisper library.

5. **Command Processing:** The transcribed text is sent to the ChatGPT node, which generates a response and determines the action.

6. **Response and Action:**

    - If the action is **1**, the conversation is expected to continue, and the system starts listening again.

- If the action is **0**, the conversation ends, and a "resume work" message is sent to the brain process.
- If the action is **2**, the robot is instructed to move to a specific location based on the response.

Examples of actions include:

- `["going to location A. please follow me", 2, "A"]`
- `["I couldn't hear you properly. please repeat what you said", 1, 0]`
- `["Good day to you too!", 0, 0]`

The system continues responding based on the conversation history until the action is '0'. When the action is '0', the conversation history is cleared.

The detailed workflow is illustrated in the following flowchart:

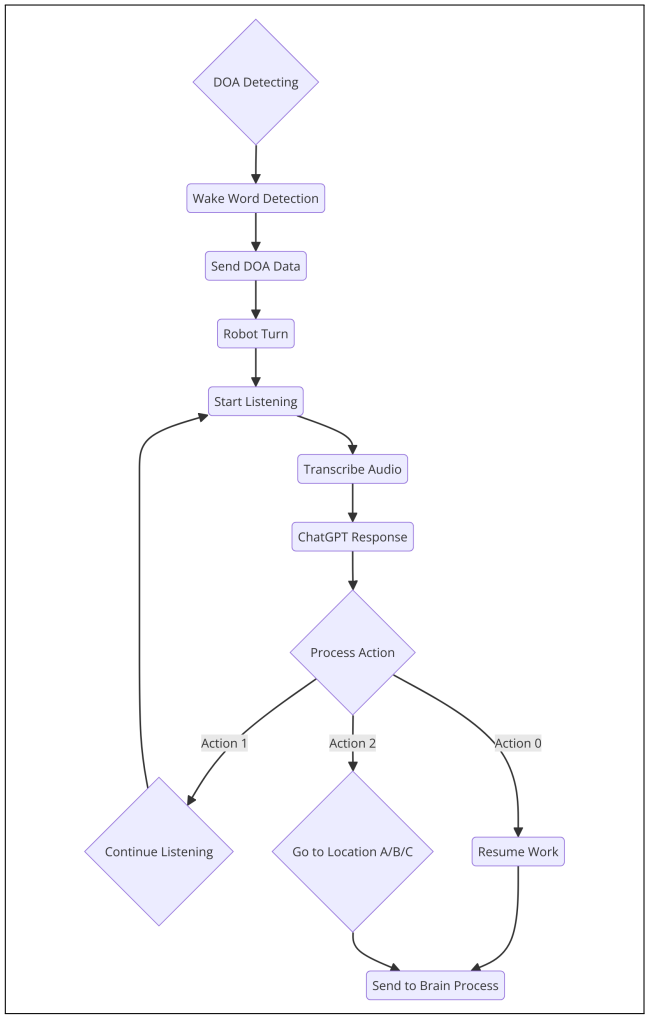



Figure 5.1: Voice Interaction System Workflow

This integration ensures that the TOTO robot can effectively and accurately respond to voice commands, providing a robust and interactive user experience.

## 5.4 Challenges and Solutions

Implementing the voice interaction system for the TOTO robot posed several challenges, each requiring specific solutions to ensure seamless operation and accurate responses. This section outlines the primary challenges encountered and the solutions implemented to address them.

### 5.4.1 High CPU Usage in Speech Transcription

One of the primary challenges encountered was the high CPU usage during the speech transcription process. The Whisper library, while effective in transcribing speech, consumed a significant amount of computational resources, impacting the overall performance of the robot.

#### Initial Solution: Predefined Commands

As a temporary solution, the transcription process was optimized to focus on predefined commands rather than general voice recognition. This approach reduced the computational load, allowing for more efficient processing and improved system responsiveness. However, this solution was not ideal for a robust and professional implementation.

#### Benchmarking and Optimization

To find a more permanent solution, the performance of the Whisper library was benchmarked using different systems, including an Intel NUC and a

Raspberry Pi 4. The benchmarking involved testing various thread counts and context sizes to identify the most efficient configuration for the Raspberry Pi 4, which is the main computing platform for the TOTO robot.

The benchmarking data for the Raspberry Pi 4 showed the following results:

| **Number of Threads** | **Tiny Model [ms]** | **Base Model [ms]** |
|---|---|---|
| 1 | 13059.40 | 28282.50 |
| 2 | 6532.99 | 14032.94 |
| 3 | 4658.64 | 9809.37 |
| 4 | 3808.86 | 7876.67 |

Table 5.1: Performance of Whisper Library on Raspberry Pi 4 with Different Thread Counts

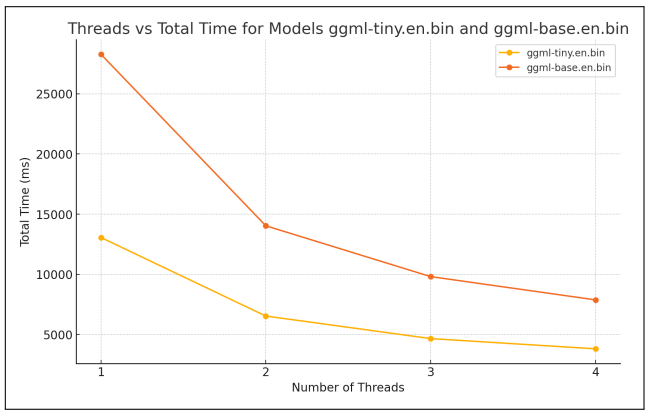



Figure 5.2: Voice Interaction System Workflow

The benchmarking indicated that using 4 threads provided the best balance between performance and resource usage. Additionally, the context size of the Whisper library was adjusted to further optimize the processing time. The following table shows the effect of different audio context sizes on the

total processing time and accuracy:

| **Audio Context Size** | **Total Time (ms)** | **Accuracy** |
|---|---|---|
| 0 | 7823.54 | Accurate |
| 128 | 114031.17 | Very inaccurate |
| 256 | 4125.93 | Partially accurate |
| 384 | 29552.75 | Partially accurate |
| 512 | 20911.6 | Accurate |
| 640 | 3620.18 | Accurate |
| 768 | 4153.97 | Accurate |
| 896 | 4760.38 | Accurate |
| 1024 | 5374.63 | Accurate |
| 1152 | 6004.27 | Accurate |
| 1280 | 6675.25 | Accurate |

Table 5.2: Effect of Audio Context Size on Processing Time and Accuracy

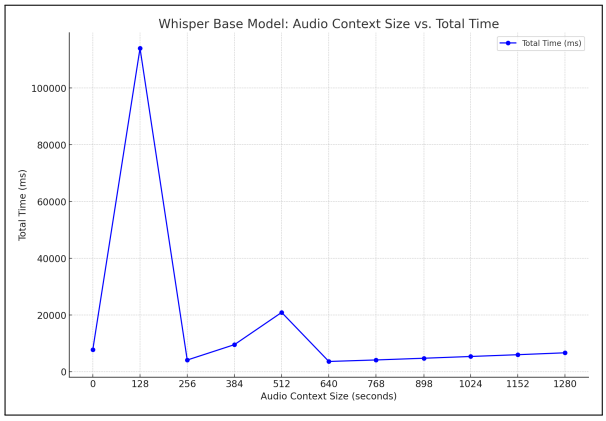



Figure 5.3: Voice Interaction System Workflow

Based on the benchmarking results, the Whisper library was configured to use 4 threads and an optimal context size of 512 to achieve efficient and accurate transcription with minimal CPU usage.

### 5.4.2 Microphone Integration and Compatibility

Integrating the HD microphone with the Raspberry Pi posed compatibility issues, particularly because the 3.5mm jack on the Raspberry Pi was an output pin, not an input.

**Solution:** The integration was successfully achieved by using the USB interface for the HD microphone. The Alsa library was utilized to manage audio input, ensuring reliable data capture for the VAD and transcription processes.

### 5.4.3 Building and Integrating VAD Libraries

Building and integrating various VAD libraries, such as WebRTC-VAD and CMU Sphinx, presented significant challenges due to compatibility issues and the complexity of the libraries.

**Solution:** After extensive research and testing, the Whisper library was selected for its robustness and compatibility with ROS2. This library provided accurate voice activity detection and seamless integration, facilitating real-time audio processing.

### 5.4.4 Directional Orientation and Accuracy

Accurately determining the direction of arrival (DOA) of voice commands was crucial for orienting the robot towards the speaker. Ensuring precise orientation and responsiveness posed a challenge.

**Solution:** The DOA system was integrated with the robot's movement control, enabling it to calculate the target angle based on the direction information and adjust its orientation accordingly. This ensured that the robot could accurately face the speaker before processing commands.

### 5.4.5 Wake Word Detection Accuracy

Ensuring high accuracy in wake word detection was essential to avoid false triggers and ensure the system responded only when necessary.

**Solution:** The Porcupine library was chosen for its high accuracy and low computational requirements. This library effectively detected predefined wake words, minimizing false triggers and optimizing system performance.

## 5.5 Testing and Results

The voice interaction system was subjected to rigorous testing to evaluate its performance, accuracy, and responsiveness. This section presents the testing methodologies, results, and insights gained from the testing phase.

### 5.5.1 Testing Methodologies

Testing involved both simulated environments and real-world scenarios to ensure comprehensive evaluation. The following methodologies were used:

- **Simulated Environment:** Initial testing was conducted in a simu-

lated environment using Gazebo and RViz to evaluate system performance and identify potential issues before real-world implementation.

- **Real-World Scenarios:** The system was tested in various real-world scenarios to assess its accuracy and responsiveness in dynamic environments.

- **Component-Specific Testing:** Each component (VAD, WWD, transcription, command processing, and TTS) was tested individually to ensure they functioned correctly before full system integration.

### 5.5.2 Results and Analysis

**Voice Activity Detection:** The Whisper library provided accurate detection of voice activity, effectively distinguishing between speech and non-speech segments.

**Wake Word Detection:** The Porcupine library demonstrated high accuracy in detecting predefined wake words, with minimal false triggers.

**Speech Transcription:** The Whisper library effectively transcribed speech to text, although initial high CPU usage was mitigated through optimization.

**Command Processing:** The ChatGPT model accurately interpreted transcribed commands and generated appropriate responses, enhancing the robot's interaction capabilities.

**Text-to-Speech:** The OpenAI TTS model produced high-quality, natural-sounding speech, improving the overall user experience.

**Directional Orientation:** The DOA system accurately determined the direction of voice commands, enabling the robot to orient itself towards the speaker effectively.

### 5.5.3 Insights and Improvements

The testing phase provided valuable insights into the system's performance and areas for improvement. Key insights included:

- Optimizing transcription processes significantly reduced CPU usage, improving overall system performance.
- Ensuring compatibility and seamless integration of components was crucial for robust system operation.
- Continuous monitoring and adjustment of parameters enhanced the accuracy and responsiveness of the system.

## 5.6 Conclusion

- Successful integration of VAD, WWD, transcription, command processing, and TTS components.
- Optimization of system components to improve performance and reduce CPU usage.

- Accurate directional orientation and voice command processing.
- Development of a dedicated module for the robot with a DOA system, microphone, and Raspberry Pi, attached to the robot. The final module is shown in Figure 5.4.

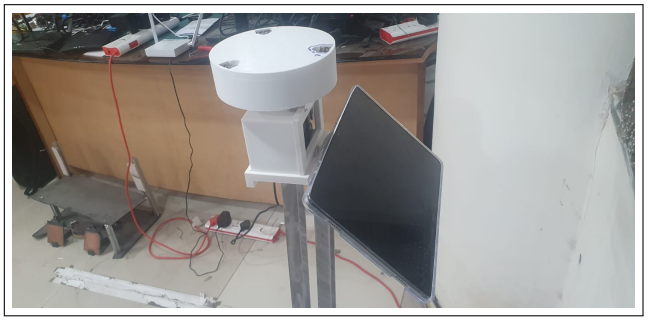


Figure 5.4: Final module of the robot with DOA system, microphone, and Raspberry Pi

# Chapter 6

# Conclusion

## 6.1 Overview

This chapter summarizes the key achievements and learning experiences during my internship under Prof. S K Saha and the mentorship of Mr. Srikrishna S , Director of Seianmai Technologies. The internship involved significant work on enhancing the TOTO robot's capabilities, including anti-crash logic, parameter tuning, autonomous mapping, and voice interaction systems. Each of these projects aimed to improve the robot's efficiency, safety, and user interaction, contributing to the broader goals of the project.

## 6.2 Major Works and Outcomes

Throughout the internship, several critical milestones were achieved, showcasing the integration of theoretical knowledge with practical implementation:

### 6.2.1 Learning and Setting up ROS2 Environment

One of the initial tasks was to familiarize myself with ROS2 and set up the environment for the TOTO robot. This included understanding the existing TOTO RE framework, learning to control, navigate, and map the environment with the robot. This foundational knowledge was crucial for executing subsequent tasks effectively.

### 6.2.2 Anti-Crash Logic Development

The development of anti-crash logic significantly enhanced the robot's operational safety. By dynamically adjusting the robot's trajectory and speed in response to detected obstacles, we ensured that the robot could navigate various environments without collisions. This was particularly important for real-world scenarios where unpredictable obstacles could pose a risk.

### 6.2.3 Parameter Tuning for Improved Navigation

Extensive parameter tuning was performed to improve the robot's navigation and obstacle avoidance capabilities. Adjustments made to the AMCL, behavior tree navigator, controller server, and costmap configurations resulted in more robust and smooth navigation performance. This process was crucial in ensuring that the robot could reach its goals accurately and efficiently while avoiding dynamic obstacles.

### 6.2.4 Autonomous Mapping Implementation

The implementation of the wave frontier algorithm for autonomous mapping allowed the robot to explore unknown environments and create detailed maps autonomously. While the initial implementation was successful in simulations, real-world testing revealed areas for further optimization and improvement.

### 6.2.5 Voice Interaction System Integration

The voice interaction system significantly enhanced the TOTO robot's user interaction capabilities. By integrating voice activity detection (VAD), wake word detection (WWD), speech transcription, command processing using ChatGPT, and text-to-speech (TTS) response generation, the robot could respond to voice commands accurately and efficiently.

## 6.3 Challenges and Solutions

Throughout the internship, several challenges were encountered, each presenting unique learning opportunities:

### 6.3.1 High CPU Usage in Speech Transcription

Initially, the Whisper library's high CPU usage posed a significant challenge. To address this, benchmarking and optimization were performed, focusing

on predefined commands and adjusting thread counts and context sizes to find the most efficient configuration for the Raspberry Pi 4.

### 6.3.2 Microphone Integration and Compatibility

Integrating the HD microphone with the Raspberry Pi required addressing compatibility issues. Using the USB interface and Alsa library for audio input management ensured reliable data capture for the VAD and transcription processes.

### 6.3.3 Wave Frontier Algorithm Optimization

While the wave frontier algorithm was effective in simulations, real-world implementation revealed issues such as small gaps being considered frontiers and high computational demands. These challenges highlighted the need for further optimization and balancing between navigation, SLAM, and mapping processes.

## 6.4 Learning Experiences

The internship provided numerous learning experiences that were instrumental in my professional development:

### 6.4.1 ROS2 and TOTO RE Framework

I learned about ROS2, how to set up the environment, and how to work with the TOTO RE framework, including controlling, navigating, and mapping the robot.

### 6.4.2 Subtasks and Implementations

- Filtering the robot frame from being displayed during mapping by testing RPLidar capabilities.
- Running SLAM standalone to reduce computational load while mapping.
- Learning about Raspberry PICO, microROS, and starting PID tuning.
- Implementing a battery percentage calculation algorithm based on voltage data.
- Researching and attempting to implement Life Long mapping.[19]
- Exploring dynamic environment changes and managed nodes in ROS2 for multiple robot communication.[21]
- Switching to Cyclone DDS for better communication in ROS2.[22]
- Creating a node for controlling other nodes based on message topics from the UI.

### 6.4.3 Memorable Experiences

Attending the Startup Mahakumbh was one of the most memorable experiences, providing an opportunity to showcase our work and interact with industry professionals.


Figure 6.1: Attending the Startup Mahakumbh

## 6.5 Conclusion

In conclusion, this internship has been an incredibly enriching experience, providing hands-on learning and significant advancements in the TOTO robot's capabilities. The integration of anti-crash logic, parameter tuning, autonomous mapping, and voice interaction systems has made the robot more efficient, safe, and user-friendly. The challenges encountered and the solutions implemented have equipped me with valuable skills and knowledge, laying a strong foundation for future work in robotics and automation.

# Bibliography

[1] Official Website of Seianmai Technologies, Available at: https://seianmai.tech/.

[2] Navigation 2 Documentation: Tuning Guide, Available at: https://docs.nav2.org/tuning/index.htm.

[3] Navigation 2 Documentation: Getting Started, Available at: https://docs.nav2.org/getting_started/index.html.

[4] M. Shah, U. Patel, N. Patel, and H. Patel, "Mapping and Navigation of Autonomous Robot with Lidar for Indoor Applications," ResearchGate, 2023. Available at: https://www.researchgate.net/publication/375893446_MAPPING_AND_NAVIGATION_OF_AUTONOMOUS_ROBOT_WITH_LIDAR_FOR_INDOOR_APPLICATIONS.

[5] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in IEEE International Conference on Robotics and Automation, vol. 1, pp. 476-481, 2000. Available at: https://ieeexplore.ieee.org/document/182671.

[6] M. Shah, U. Patel, N. Patel, and H. Patel, "ROS based Compact Mobile Robot for Area Mapping, Autonomous Navigation and Path Planning," ResearchGate, 2023. Available at: https://www.researchgate.net/publication/375895922_ROS_based_Compact_Mobile_Robot_for_Area_Mapping_Autonomous_Navigation_and_Path_Planning.

[7] S. Bhattacharya and S. Ghosh, "Autonomous Indoor Mobile Robot Exploration Based on Wavefront Algorithm," in International Conference on Mechatronics and Robotics Engineering, vol. 1, pp. 53-58, 2019. Available at: https://www.researchgate.net/publication/334983056_Autonomous_Indoor_Mobile_Robot_Exploration_Based_on_Wavefront_Algorithm.

[8] M. Wiseman, "py-webrtcvad," GitHub repository. Available at: https://github.com/wiseman/py-webrtcvad.

[9] The Click Reader, "WebRTC Voice Activity Detection using Python," Medium article. Available at: https://medium.com/@theclickreader/webrtc-voice-activity-detection-using-python-the-click-reader-9ee3797adbea.

[10] CMU Sphinx, "Voice Activity Detection," CMU Sphinx Wiki. Available at: https://cmusphinx.github.io/wiki/asr/vad/.

[11] CMU Sphinx, "Discussion on Sphinx4," SourceForge. Available at: https://sourceforge.net/p/cmusphinx/discussion/sphinx4/thread/d27567b3/.

[12] CMU Sphinx, "Pocketsphinx Documentation." Available at: https://cmusphinx.github.io/doc/pocketsphinx/index.html.

[13] Picovoice, "Porcupine Wake Word Detection." Available at: https://picovoice.ai/platform/porcupine/.

[14] OpenAI, "Whisper," GitHub repository. Available at: https://github.com/openai/whisper.

[15] G. Gerganov, "whisper.cpp," GitHub repository. Available at: https://github.com/ggerganov/whisper.cpp.

[16] OpenAI, "Platform Documentation: Overview." Available at: https://platform.openai.com/docs/overview.

[17] OpenAI, "Text-to-Speech Guide." Available at: https://platform.openai.com/docs/guides/text-to-speech.

[18] Boston Dynamics, "Robots That Can Chat." Available at: https://bostondynamics.com/blog/robots-that-can-chat/.

[19] S. Macenski, "slam_toolbox," GitHub repository. Available at: https://github.com/SteveMacenski/slam_toolbox.

[20] S. Macenski, "Life Long Mapping," YouTube video. Available at: https://www.youtube.com/watch?v=rZOxPGCn4QM.

[21] ROS2 Design, "Managed Nodes," ROS2 Design Articles. Available at: https://design.ros2.org/articles/node_lifecycle.html.

[22] ROS2 Documentation, "DDS Implementations in ROS2." Available at: https://docs.ros.org/en/humble/Installation/DDS-Implementations.html.

[23] ROS Discourse, "New Fast DDS Performance Testing." Available at: https://discourse.ros.org/t/new-fast-dds-performance-testing/29539.

[24] eProsima, "Fast DDS vs Cyclone DDS Performance." Available at: https://www.eprosima.com/index.php/resources-all/performance/fast-dds-vs-cyclone-dds-performance.