Capstone Project I Report

Optimizing SLAM Algorithms for Mobile Robots

Alma Toleubekova, Aidana Zhumagaliyeva, Kir Smolyarchuk

A thesis submitted in part fulfilment of the degree of

BS in Robotics Engineering

Supervisor: Ton Duc Do

Instructor: Anara Sandygulova

Department of Robotics and Mechatronics

School of Engineering and Digital Sciences

Nazarbayev University

May 5, 2025

Table of Contents

Abstract

The ability of mobile robots to autonomously navigate and interact with their surroundings is critical across various domains, including industrial automation, service robotics, and autonomous vehicles. A fundamental challenge in achieving autonomy lies in enabling robots to simultaneously construct a map of an unknown environment while localizing themselves within it — a problem known as Simultaneous Localization and Mapping (SLAM). Accurate environmental mapping is essential for tasks such as path planning, obstacle avoidance, and high-level decision-making, allowing robots to operate effectively in dynamic and unstructured settings.

This research initially focused on enhancing SLAM performance by integrating 2D LiDAR (RPLiDAR A1) and a depth camera (Intel RealSense D435i) on a TurtleBot3 Burger within the ROS Noetic framework. By fusing LiDAR and visual depth data, we aimed to improve the accuracy and robustness of 3D point cloud-based environmental representations. A comparative analysis of SLAM methodologies—including FastSLAM, GraphSLAM, ORB-SLAM, LiDAR-based SLAM, and visual SLAM—was conducted to assess their suitability for real-time mobile robot navigation.

Building upon the SLAM framework, the second phase of this study focuses on sampling-based motion planning techniques, specifically Rapidly-exploring Random Trees (RRT) and its variants, to enable autonomous navigation within the constructed 3D map. To further evaluate the scalability and practical deployment of these algorithms, we are transitioning our research to a larger autonomous mobile robot platform equipped with enhanced sensing capabilities and computational resources. This work aims to develop a comprehensive mapping and navigation framework that optimizes both localization accuracy and motion efficiency for real-world robotic applications.

Chapter 1: **Introduction**

Simultaneous Localization and Mapping (SLAM) is a foundational capability in autonomous robotics. It enables a mobile robot to build a map of an unknown environment while simultaneously estimating its own position within that map. SLAM is essential for a wide range of real-world applications, including autonomous vehicles, warehouse automation, service robots in dynamic home or office settings, and exploration robots operating in GPS-denied environments such as underground tunnels or disaster zones.

Despite significant progress, implementing SLAM in real-world, dynamic, and unstructured environments remains a major challenge. This is due to several factors: the accuracy limitations of individual sensors (e.g., LiDAR or cameras alone), difficulties in achieving real-time performance on resource-constrained platforms, and the lack of robustness in handling changes in lighting, occlusion, and motion. Traditional SLAM systems often use single-sensor data, making them prone to failure when that sensor is compromised. For instance, 2D LiDAR may perform well on flat indoor floors but lacks elevation information, while depth cameras may struggle in outdoor lighting conditions or long distances.

The core research problem addressed in this project is: *How can we design and implement a scalable, real-time SLAM system that improves localization accuracy and map quality by fusing complementary sensor data from LiDAR, depth cameras, and IMUs, especially in complex, dynamic environments?*

This project explores a sensor-fusion-based SLAM system that leverages the strengths of multiple sensing modalities. Specifically, we integrate 2D LiDAR, the Intel RealSense D435i RGB-D camera (which includes an inertial measurement unit), and simulate advanced environments using Gazebo. The sensor data is fused using filtering techniques, such as the Extended Kalman Filter (EKF), to reduce uncertainty and enhance the precision of robot localization and environmental mapping.

The project is organized into three progressive stages to investigate, develop, and evaluate this integrated SLAM system:

- **Stage 1: 2D SLAM and Navigation with TurtleBot3 Burger and RPLiDAR**
  In this foundational phase, we employ a cost-effective, open-source mobile robot platform—the TurtleBot3 Burger—equipped with a 2D RPLiDAR sensor. The focus here is on understanding and implementing classical 2D SLAM algorithms such as GMapping and HectorSLAM using ROS (Robot Operating System). Through this process, we establish essential knowledge of real-time mapping and autonomous navigation, while also addressing practical challenges such as loop closure, localization drift, and sensor noise.
- **Stage 2: 3D Point Cloud Mapping and Sensor Fusion with RealSense D435i and LiDAR**
  Recognizing the limitations of 2D SLAM in capturing the depth and structure of real-world environments, this stage extends the system by integrating the Intel RealSense D435i RGB-D camera. The camera provides both color (RGB) and depth information, along with an onboard IMU. Combining these inputs with 2D LiDAR data allows us to build more detailed 3D point cloud maps and implement sensor fusion techniques for better pose estimation. This stage focuses on 3D mapping tools such as RTAB-Map

and OctoMap, and tackles challenges related to data synchronization, calibration, and real-time processing.

- **Stage 3: 3D Mesh Mapping and Path Planning in Simulated Gazebo Environment**
  The final stage involves evaluating the scalability and adaptability of the developed SLAM system in a controlled virtual environment. Using the Gazebo simulator, we create diverse terrain scenarios to simulate complex navigation tasks. The objective is to assess how the fused SLAM system performs in larger, multi-level, or cluttered environments and to implement path planning strategies based on the generated 3D maps. This allows us to safely test advanced algorithms such as frontier exploration and 3D obstacle avoidance before deployment on physical hardware.

Motivation: This work is motivated by the need for robust, cost-effective SLAM systems that can operate reliably in real-world environments with limited infrastructure support. In Kazakhstan and other developing robotics markets, scalable and open-source solutions are essential for accelerating technological adoption. By systematically exploring SLAM from simple to complex settings, and by integrating affordable sensor technologies, this project offers a practical roadmap for researchers, students, and developers working in mobile robotics.

Applications and Use Cases: The outcomes of this research can be used in various domains, such as:

- **Warehouse automation** — Robots navigating tight, dynamic layouts for inventory management.
- **Search and rescue** — Autonomous agents mapping partially collapsed buildings or underground areas.
- **Home and office robotics** — Smart service robots that adapt to changes in indoor environments.
- **Education and research** — Modular systems that can be adopted by universities or startups for prototyping and testing.

This project also contributes to the broader goal of fostering robotics innovation in Kazakhstan. By focusing on reproducible, low-cost, and educationally accessible methodologies, we aim to empower the next generation of engineers and scientists working toward autonomous systems in local industries.

Chapter 2: **Literature Review**

In the realm of robotics, the ability of mobile robots to navigate and interact with their environment autonomously is pivotal for their practical applicability across various domains, ranging from household assistance to industrial automation. At the heart of this autonomy lies the crucial task of Simultaneous Localization and Mapping (SLAM), a computational technique that enables robots to construct maps of unknown environments while concurrently determining their own position within these maps. This aspect of SLAM facilitates efficient navigation for robots, with the data generated being leveraged to make informed decisions crucial for the autonomous movement of domestic robots. The necessity for utilizing an environment map serves two primary purposes: firstly, maps are frequently required to facilitate various tasks, such as providing route plans or offering an accurate depiction of a person's surroundings; and secondly, in the absence of human intervention, robots can autonomously construct their own maps, enabling self-localization and decision-making tailored to the user's requirements [1].

There are various techniques of gazering environmental data surrounding mobile robots, such as equipping them with distance measuring tools like LiDAR, ultrasonic or infrared light (IIR) based distance sensors, utilizing image processing, or employing contact-based sensors for exploring the robot's workspace [2].

This literature review endeavors to provide a comprehensive overview of the existing methodologies, techniques, and algorithms employed in SLAM for mobile robots, with a specific focus on identifying the optimal algorithmic approaches [3]. By synthesizing insights from seminal research works and cutting-edge developments, this review aims to elucidate the key challenges, advancements, and trends in the field of SLAM, thereby laying the groundwork for the subsequent phases of our project.

To contextualize the discussion, it is imperative to delve into the historical evolution of SLAM and trace its trajectory from its nascent stages to its current state-of-the-art implementations. Furthermore, a critical examination of the underlying principles and mathematical frameworks governing SLAM algorithms is essential for discerning the strengths and limitations of different approaches [4]. Additionally, insights garnered from real-world deployments and experimental evaluations play a pivotal role in gauging the efficacy and performance of various SLAM techniques across diverse operational scenarios [5].

Therefore, taking into account the importance of SLAM in control of mobile robots, this paper aims to analyze common approaches of integrating SLAM and, considering factors like specific requirements, sensor and environmental characteristics, choose the most suitable technique for a particular machine.

This paper will be organized as follows: section II will elaborate on fundamentals of SLAM, while section III will discuss various approaches of SLAM which are FastSLAM, GraphSLAM, ORB-SLAM, LiDAR-based SLAM, visual SLAM, Kalman Filter-based mapping and DSM. Furthermore, Challenges and Limitations of each Method will be discussed in section IV, where we will explore the techniques for improving the efficiency, accuracy, and robustness of different SLAM algorithms. Finally, we will examine further advancements and potential directions in the field of mobile robots and come to the conclusion.

Optimizing SLAM algorithms is essential for real-world applications, demanding a balance between accuracy, efficiency, and resource constraints. The review of the selected papers below delves into various optimization techniques explored in recent research:

- FastSLAM FastSLAM is a particle filter-based SLAM algorithm that efficiently handles non-linearities by representing the robot's trajectory using a set of particles and estimating the map of the environment using Bayesian methods. It offers robustness and scalability by maintaining a limited number of particles. The SLAM method for Formula Student Driverless Competition introduces analysis of FastSLAM 2.0 and GraphSLAM accuracy and performance, a particle filter-based approach that reduces computational complexity by maintaining a limited number of particles. The book by [6] expands upon FastSLAM, offering a comprehensive analysis of the algorithm and its variants. It explores optimization techniques like selective update strategies and offers the Rao-Blackwellization algorithm for improved efficiency.
  Research and exploration of SLAM solutions based on mobile robots [7] paper investigates FastSLAM 1.0 and 2.0 for mobile robot applications. It highlights the impact of particle number selection on accuracy and computational cost, emphasizing the need for optimization.
  The paper by Leonard and Feder, An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association addresses the issue of data association uncertainties in FastSLAM. It introduces an Unscented Kalman Filter (UKF) variant of FastSLAM, demonstrating improved robustness to landmark association errors.

- ORB-SLAM ORB-SLAM (Oriented FAST and Rotated BRIEF SLAM) is a feature-based SLAM algorithm that employs oriented FAST and rotated BRIEF features for real-time simultaneous localization and mapping tasks. It relies on feature detection and matching techniques, making it suitable for environments with texture-rich features but susceptible to changes in illumination and scene structure.
  A seminal contribution to SLAM research comes from Formula Student Driverless Competition with theoretical algorithms, as highlighted in various works. This original work, often referenced in the literature, serves as a cornerstone in the development of SLAM techniques tailored for specific applications. Although specific details may vary depending on the theoretical framework adopted, the research undertaken by Afonso Luís, Correia Nazaré, Lopes Pedro Daniel, and others remains pivotal in shaping the discourse surrounding SLAM methodologies.[8] Furthermore, insights from [9] shed light on the principles underlying ORB-SLAM, offering a comprehensive analysis of its algorithmic variants and optimization strategies.

- A Kalman Filter-based SLAM Approach A Kalman Filter-based SLAM Approach utilizes the Kalman Filter algorithm for simultaneous localization and mapping tasks. It relies on fusing sensor measurements with a state estimator to estimate the robot's pose and the map of the environment, offering efficiency but limited by assumptions of linearity and Gaussian noise.
  A Kalman Filter-based SLAM Approach [10] work proposes an Extended Kalman Filter (EKF) based SLAM approach, offering an alternative to particle filters. The paper discusses the computational efficiency of EKF compared to particle filters, but also acknowledges its limitations in handling non-linearities. Kalman Filters for Robotics [11]book serves as a reference for Kalman filter applications in robotics, including EKF-based SLAM. Understanding Kalman filters is essential for optimizing EKF-SLAM algorithms.

- Rao-Blackwellized Particle Filters Rao-Blackwellized Particle Filters combine the strengths of particle filters and Kalman filters in SLAM tasks. They efficiently handle uncertainty by employing a particle filter for robot pose estimation and a Kalman filter for landmark estimation, offering robustness but requiring parameter tuning and complex implementation.
  Heuristic Optimization for Efficient SLAM Using Rao-Blackwellized Particle Filters work explores Rao-Blackwellization for SLAM optimization. It proposes heuristic techniques to selectively update robot pose and landmark estimates, reducing computational load without compromising accuracy.
  Leonard and Feder [12] in his work investigated the Rao-Blackwellization for multi-robot SLAM. It demonstrates the potential for significant computational savings through this approach.

- Decoupled Stochastic Mapping (DSM) Decoupled Stochastic Mapping (DSM) is an innovative SLAM approach that separates the mapping and localization processes, reducing computational complexity. It employs optimization techniques to split SLAM problems into back-end and front-end tasks, making it suitable for large-scale mapping with reduced computational resources.

Real scenario testing for mapping, localization, and data association performance [9] book expands upon DSM, offering a comprehensive analysis of the algorithm and its variants. It explores optimization techniques like selective update strategies and Rao-Blackwellization for improved efficiency. Research and exploration of SLAM solutions based on mobile robots [13] paper investigates DSM for mobile robot applications. It highlights the impact of particle number selection on accuracy and computational cost, emphasizing the need for optimization. An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association [14] paper addresses the issue of data association uncertainties in DSM. It introduces an Unscented Kalman Filter (UKF) variant of DSM, demonstrating improved robustness to landmark association errors.
Presented below is a comparative analysis table elucidating the strengths and weaknesses of notable SLAM algorithms, namely FastSLAM, ORB-SLAM, Kalman Filter-based SLAM, Rao-Blackwellized Particle Filter and Decoupled Stochastic Mapping (DSM). By delineating their respective advantages and disadvantages, this table aims to provide a comprehensive understanding of the performance characteristics inherent in each algorithm, aiding researchers and practitioners in making informed decisions for SLAM applications.

Determining the most suitable algorithm depends on the specific requirements of the application, including computational resources, sensor capabilities, environmental characteristics, and desired trade-offs between accuracy and efficiency.

The outlook for SLAM in mobile robotics entails several key directions based on recent research findings. Integrating deep learning for feature extraction in visual SLAM represents a significant trend with potential performance enhancements. Moreover, the utilization of edge computing for real-time processing, as discussed in [15], offers opportunities for improving system efficiency. However, challenges persist, particularly in adapting SLAM strategies for dynamic environments. Similarly, scalability remains an ongoing concern, necessitating further exploration of decentralized architectures. From a practical standpoint, insights from SLAM research inform the development of user-friendly interfaces and regulatory considerations, emphasizing the importance of equitable access and privacy protection. As for the cross-disciplinary opportunities, collaboration with computer vision and human-computer interaction fields holds promise for revolutionizing autonomous navigation motion planning.

Algorithm	Advantages	Disadvantages
FastSLAM	- Handles non-linearities effectively. - Efficient in dynamic environments.	- Computational complexity increases with the number of particles.
ORB-SLAM	- Fast feature detection and matching techniques for real-time performance.	- Sensitive to changes in illumination and scene structure.
Kalman Filter-based SLAM	- Efficient fusion of sensor measurements for real-time estimation.	- Assumes linear dynamics and Gaussian noise, limiting applicability in complex environments.
Rao-Blackwellized Particle Filter	- Combines the strengths of particle filters and Kalman filters. - Efficiently handles uncertainty in SLAM problems.	- Requires tuning of parameters for optimal performance. - Complex to implement and understand.
Decoupled Stochastic Mapping (DSM)	- Decouples mapping and localization processes, reducing computational complexity. - Suitable for large-scale mapping.	- May require additional computational resources for maintaining map consistency.

Table 2.1: Advantages and Disadvantages of the SLAM algorithms

The comparison of various methods for Simultaneous Localization and Mapping (SLAM) is crucial for identifying the strengths and limitations of each approach, ultimately guiding the selection of the most suitable method for specific applications. In this literature review, we have examined several prominent SLAM techniques, including FastSLAM, ORB-SLAM, Kalman Filter-based SLAM, Rao-Blackwellized Particle Filter, and Decoupled Stochastic Mapping (DSM).

FastSLAM emerges as a robust option, particularly adept at handling non-linearities and dynamic environments. Its efficiency in dynamic scenarios is commendable, allowing for accurate localization and mapping even amidst rapid changes. However, the computational complexity of FastSLAM scales with the number of particles, which can pose challenges in resource-constrained environments or applications requiring real-time performance. ORB-SLAM offers notable advantages in terms of fast feature detection and matching techniques, ensuring real-time performance. However, its sensitivity to changes in illumination and scene structure might limit its applicability in environments with varying lighting conditions or scene dynamics.

Kalman Filter-based SLAM stands out for its efficient fusion of sensor measurements, facilitating real-time estimation. Nevertheless, its reliance on linear dynamics and Gaussian noise assumptions could restrict its effectiveness in complex and highly dynamic environments where non-linearities and uncertainties are prevalent.

Rao-Blackwellized Particle Filter represents a promising fusion of particle filters and Kalman filters, effectively leveraging their respective strengths. While capable of efficiently handling uncertainty in SLAM problems, its implementation complexity and the need for parameter tuning may deter widespread adoption, particularly in applications where simplicity and ease of implementation are paramount.

Decoupled Stochastic Mapping (DSM) addresses computational complexity by decoupling mapping and localization processes, making it suitable for large-scale mapping tasks. However, maintaining map consistency may necessitate additional computational resources, potentially impacting its scalability and real-time performance.

In summary, each SLAM method offers distinct advantages and disadvantages, catering to different requirements and scenarios. The selection of an appropriate SLAM technique should consider factors such as computational resources, real-time performance constraints, environmental dynamics, and the level of accuracy required. Future research efforts may focus on overcoming the limitations of existing methods and developing hybrid approaches that combine the strengths of multiple techniques to achieve enhanced performance and versatility in SLAM applications.

Chapter 3: **Methodology**

The primary objective of our project is to enhance SLAM performance by integrating multiple sensors and leveraging optimized algorithms. Our methodology is structured into distinct phases, each focusing on incremental improvements to ensure a robust and adaptable system capable of functioning across different robotic platforms.

The system is designed to integrate a variety of sensors, including a 2D LiDAR, a 3D depth camera, and additional sensors like radar and IMU, to gather comprehensive environmental data. To merge the diverse data streams effectively, sensor fusion techniques, including Kalman filters, are employed to improve the reliability and accuracy of the SLAM process. The methodology is implemented in the following stages:

3.1 Stage 1: 2D SLAM and Navigation with TurtleBot3 Burger and RPLiDAR

The first stage of the project focuses on establishing a foundation for SLAM by using a cost-effective, open-source robot platform – the TurtleBot3 Burger – paired with a 2D RPLiDAR sensor. This phase is critical as it allows us to understand the fundamental principles of mapping and localization using ROS (Robot Operating System), while also familiarizing ourselves with the challenges of real-time robotic navigation.

- **Hardware Setup:** The TurtleBot3 Burger, a compact differential-drive robot, is equipped with an RPLiDAR A1, a 2D laser scanner capable of capturing 360-degree horizontal scans. This configuration allows the robot to detect surrounding obstacles and build a two-dimensional representation of its environment.
- **SLAM Algorithms:** Several widely-used 2D SLAM algorithms are tested, including Gmapping, Hector SLAM, and Cartographer. These algorithms are implemented through ROS packages and evaluated based on their ability to construct accurate occupancy grid maps. Each algorithm has different characteristics:
  - **Gmapping** relies on particle filters and is well-suited for moderate motion noise.
  - **Hector SLAM** uses scan matching and does not require odometry input, making it useful for scenarios with poor wheel encoder data.
  - **Cartographer** supports real-time loop closure and provides highly accurate maps but has higher computational requirements.
- **Mapping and Evaluation:** The robot is manually or autonomously navigated in a structured indoor environment to generate static maps. Later, dynamic elements (e.g., moving objects) are introduced to assess the adaptability and robustness of each algorithm. Performance is evaluated in terms of:
  - Map completeness and resolution.
  - Localization accuracy and drift.

Table 3.1: Characteristics of Selected 2D Lidar SLAM Algorithms

Name	Characteristic
EKF SLAM	Capable of building a sparse map using features, but with a high computational cost and limited robustness.
Gmapping	Based on the particle filter, mitigates particle degradation but relies on odometry; lacking loop detection, only suitable for small-scale scene construction.
Hector SLAM	The pose can be constructed in real time without an odometer. Sensitive to initial value, lacking loop detection, difficult to guarantee accuracy, applicable to air or flat-road environments, drifts with fast rotation.
Cartographer	A classic diagram optimization SLAM, applicable to 2D/3D radar, with the front end adopting CSM and gradient optimization and the back end adopting graph optimization, with accelerated closed-loop detection. Strong real-time performance, applicable to large scenes.
CNN SLAM	The deep learning method can adapt to various situations and improve accuracy and robustness, but it requires a large amount of data-scene changes, which reduces detection accuracy.

  - Real-time processing efficiency.

- **Navigation Stack Integration:** Once reliable 2D maps are created, we integrate the ROS Navigation Stack. This includes global path planning using algorithms like Dijkstra or A*, and local obstacle avoidance using the Dynamic Window Approach (DWA). The robot's ability to reach user-defined goals while avoiding static and dynamic obstacles is validated.

3.2 Stage 2: 3D Point Cloud Mapping and Sensor Fusion with RealSense D435i and LiDAR

While 2D SLAM provides basic navigational capabilities, it lacks the spatial richness required for more complex environments. Therefore, Stage 2 extends the system by integrating an Intel RealSense D435i depth camera, which provides RGB and depth data, and an Inertial Measurement Unit (IMU), alongside the 2D LiDAR.

- **Objective:** The goal is to transition from 2D mapping to full 3D point cloud mapping by combining information from multiple sensors. This allows for more accurate modeling of real-world environments, including vertical structures and obstacles not captured by 2D sensors.

- **Hardware Configuration:** The RealSense D435i camera is mounted on the TurtleBot3 platform. It outputs depth images at high frequency and includes an onboard IMU for motion tracking. Data from the RealSense is synchronized with LiDAR scans and processed in a common coordinate frame using ROS TF and message time stamps.

- **Software and Algorithms:** We use RTAB-Map (Real-Time Appearance-Based Mapping), a graph-based SLAM approach capable of handling multi-modal sensor input.

It supports loop closure detection, real-time 3D mapping, and integration with point cloud libraries.

- **Sensor Fusion:** To enhance mapping robustness, sensor fusion techniques are employed. Kalman Filters, Extended Kalman Filters (EKF), and potentially Unscented Kalman Filters (UKF) are explored to combine:
  - Range data from the LiDAR for precise horizontal measurements.
  - Depth data from the camera for capturing vertical and object geometry.
  - IMU data for estimating orientation and acceleration.

  Sensor fusion improves accuracy by reducing individual sensor noise and filling in gaps where a single sensor may fail.
- **Mapping Output:** The result is a dense 3D point cloud map of the environment that captures walls, furniture, and other 3D features. This map provides richer spatial awareness for advanced navigation tasks.

3.3 Stage 3: 3D Mesh Mapping and Path Planning in Simulated Gazebo Environment

The final stage of the methodology aims to simulate advanced path planning and evaluate how well our integrated SLAM system scales in virtual environments with varying terrain complexity.

- **Objective:** Convert the 3D point cloud map from Stage 2 into a mesh-based environment and simulate robotic navigation within it. This allows the testing of various path planning strategies under controlled and repeatable conditions.
- **Mesh Generation:** Using tools like `mesh_lab`, `CloudCompare`, or RTAB-Map's built-in meshing tools, we convert the raw point cloud data into a triangular mesh. This mesh is then used to create a Gazebo-compatible world file that can simulate physical interactions and navigation.
- **Gazebo Simulation:** The TurtleBot3 robot is imported into the Gazebo simulation environment, where it navigates the 3D mesh-based world using the same 2D or hybrid planners. This step validates how well real-world maps can be translated into simulation for algorithm benchmarking.
- **2D and 2.5D Path Planning:** Initially, standard 2D global planners are evaluated. Then, we extend the investigation into 2.5D path planning. Elevation maps are generated using depth data, and planners capable of accounting for slope, surface roughness, or traversability are integrated. These include:
  - Elevation-aware costmaps.
  - Terrain analysis modules from packages such as `grid_map` or `move_base_flex`.
- **Evaluation and Output:** The simulation environment provides a safe and scalable testbed to examine:
  - Planner behavior on inclined or uneven terrain.

- Navigation robustness in large-scale environments.
- Performance of SLAM-generated maps in path execution tasks.

Methodology Summary:
This three-stage methodology reflects a systematic and scalable approach to SLAM research and implementation. Beginning with a simple 2D SLAM system and evolving toward rich 3D mapping and advanced path planning, each stage builds technical depth and real-world applicability. Sensor fusion techniques are refined at each stage to enhance robustness, while the final simulation framework provides an essential testbed for experimentation, validation, and future extensions such as multi-robot SLAM, semantic mapping, or terrain-adaptive navigation.

Chapter 4: **Progress to Date**

Over the past semester, we have made considerable progress toward our goal of optimizing SLAM algorithms for mobile robot navigation through multi-sensor data fusion. This chapter summarizes the work completed thus far, focusing on literature analysis, system setup, sensor integration, preliminary mapping, and algorithm development.

1. Literature Review and Problem Analysis

To begin the project, an extensive literature review was conducted to understand the current state-of-the-art in SLAM algorithms and sensor fusion methods. We analyzed various approaches to mapping and localization, including the use of LiDARs, depth cameras, and IMUs. The review highlighted the importance of accurate sensor data synchronization and the application of Kalman filtering techniques to reduce noise and uncertainty. Based on this analysis, we identified a gap in the effective integration of multiple sensors for enhanced SLAM performance, particularly in real-time applications involving mobile robots.

2. Integration of 2D LiDAR and Static Mapping

Our initial implementation phase involved integrating a 2D RPLIDAR sensor with the TurtleBot3 Burger platform. We utilized the HectorSLAM package in ROS Melodic to generate a static 2D occupancy grid map of an indoor environment. The robot was manually teleoperated to explore the environment, and the LiDAR continuously provided scan data, which was used to incrementally build the map. This process allowed us to test the SLAM pipeline in a controlled setting and confirm the accuracy of the generated static map.

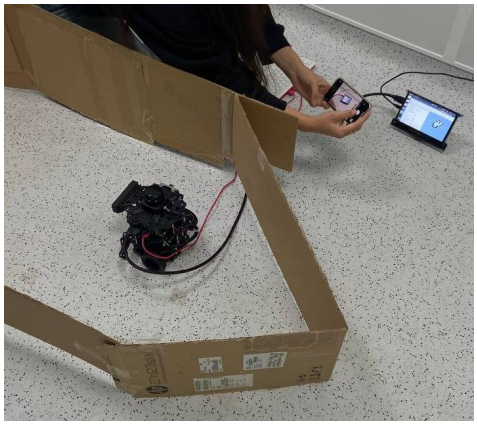


Figure 4.1: TurtleBot3 robot performing static mapping using HectorSLAM

Figure 4.2: Resulting 2D static map of the environment

3. Establishing Wireless Teleoperation and Dynamic Mapping

To enable autonomous operation without physical tethering, we established secure SSH communication between the control laptop and the TurtleBot3 Burger. This allowed us to remotely launch mapping and navigation nodes while teleoperating the robot wirelessly. The dynamic map generated during this process reflected real-time updates as the robot moved, providing more practical mapping insights compared to the initial static map. This step laid the foundation for future testing of real-time SLAM algorithms.

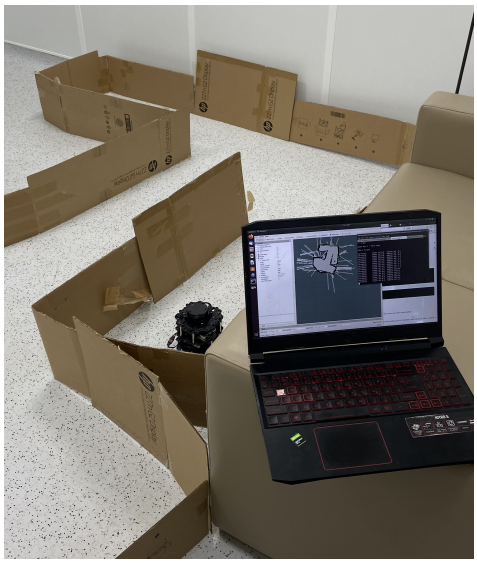


Figure 4.3: TurtleBot3 robot generating a dynamic map through wireless teleoperation

4. SLAM Tuning and Platform Upgrade

Having achieved functional mapping and localization, we turned our focus to optimizing SLAM parameters for improved performance. Specific attention was paid to tuning parameters related to update rates, scan matching, and map resolution to achieve more stable localization. To support more computationally demanding algorithms and sensor processing, we initiated a platform transition from the Raspberry Pi to the NVIDIA Jetson Nano. This upgrade allowed for better real-time performance, especially as we integrated more sensors for 3D mapping.

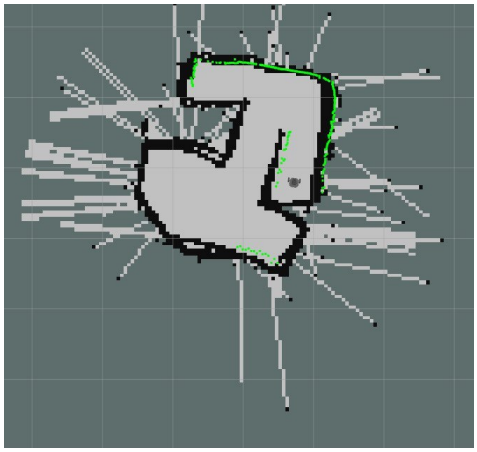


Figure 4.4: 2D occupancy grid map generated during dynamic exploration

5. 3D Mapping with OAK-D Pro Depth Camera

To move beyond 2D mapping, we integrated the OAK-D Pro depth camera, which uses stereo vision to generate real-time 3D point clouds. These point clouds were visualized and recorded to represent the environment's structure in three dimensions. This marked a critical step toward our ultimate goal of achieving full 3D SLAM capabilities.

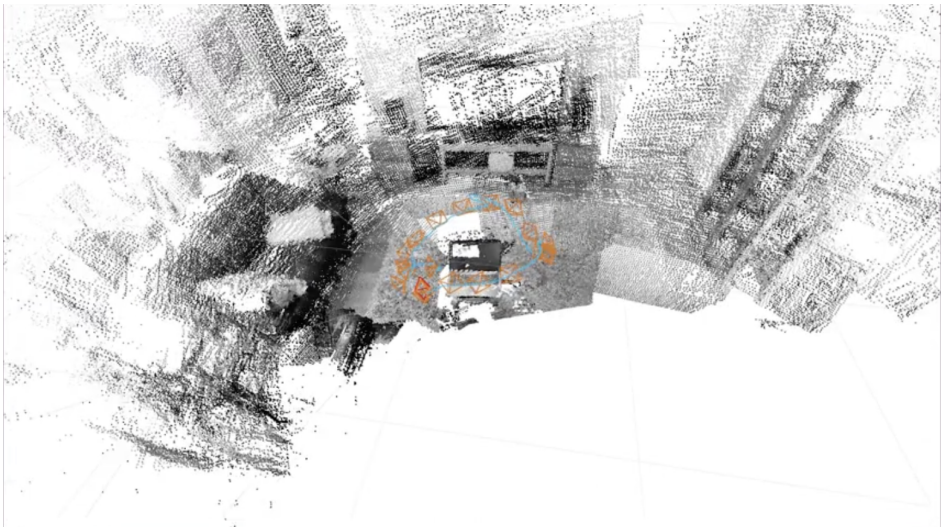


Figure 4.5: 3D point cloud map generated using the OAK-D Pro depth camera

6. Sensor Synchronization and Fusion Architecture

To perform effective sensor fusion, we designed a data architecture that allows the RPLIDAR and OAK-D Pro to operate simultaneously within ROS. We launched nodes for each sensor independently and implemented a custom TF broadcaster node to establish the appropriate static transform between their coordinate frames. This synchronization was essential for correctly fusing data into a common frame of reference for mapping and localization purposes.

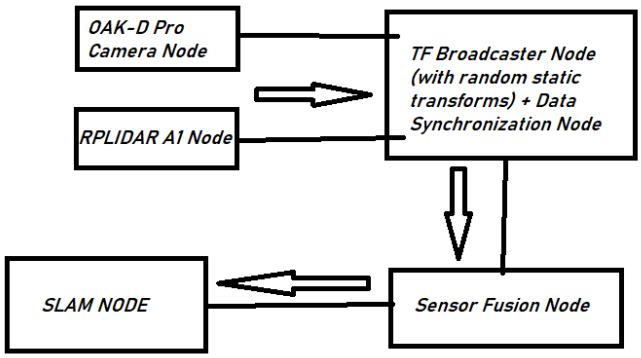


Figure 4.6: High-level architecture of sensor fusion between LiDAR and depth camera

7. Transition to Intel RealSense for Enhanced Integration

Despite the capabilities of the OAK-D Pro, we faced challenges integrating it fully with ROS navigation and mapping packages. Consequently, we transitioned to the Intel RealSense D435i, which offers better compatibility and provides both depth information and inertial data via an onboard IMU. We calibrated the RealSense data streams and launched them through the RTAB-Map framework, which enabled us to use its visual SLAM and 3D mapping capabilities. This switch significantly improved system stability and sensor data fusion.

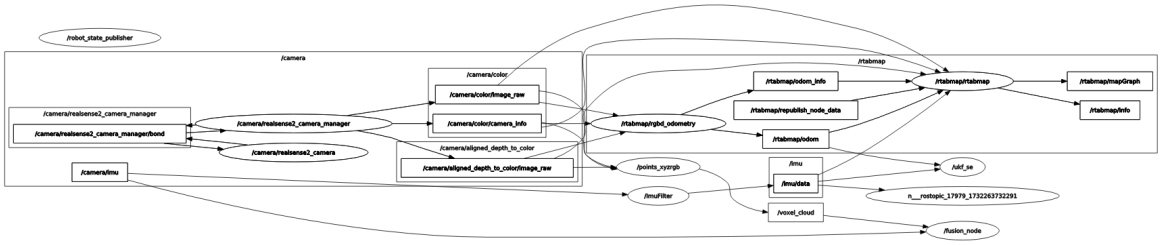


Figure 4.7: Realsense d435i Nodes

The RealSense camera system is structured under the `/camera` namespace and includes the following main components:

- `/camera/realsense2_camera_manager`: Manages the lifecycle of the RealSense device.
- `/camera/realsense2_camera`: Publishes all camera streams and sensor data.
- `/camera/color/image_raw`, `/camera/color/camera_info`: Provide raw RGB image data and camera intrinsic parameters.
- `/camera/aligned_depth_to_color/image_raw`: Publishes depth images aligned to the color frame.
- `/camera/imu`: Publishes inertial measurement data from the camera's IMU.

These outputs are consumed by nodes such as:

- `/rtabmap/rgbd_odometry`: Estimates visual odometry from synchronized RGB-D and IMU data.
- `/rtabmap/rtabmap`: Performs real-time 3D SLAM using the visual and depth information.
- `/fusion_node`: Fuses data from multiple sensors including the IMU and depth camera.

This configuration enables robust visual-inertial SLAM in real-time.

8. Sensor Fusion Results and Map Generation

After synchronizing the data streams and broadcasting the necessary transforms, we fused the data from the RPLIDAR and the RealSense depth camera. The resulting maps were richer in detail and better represented the robot's surroundings, enabling more accurate navigation. This fusion of spatial data sources led to a more comprehensive 3D environmental model.

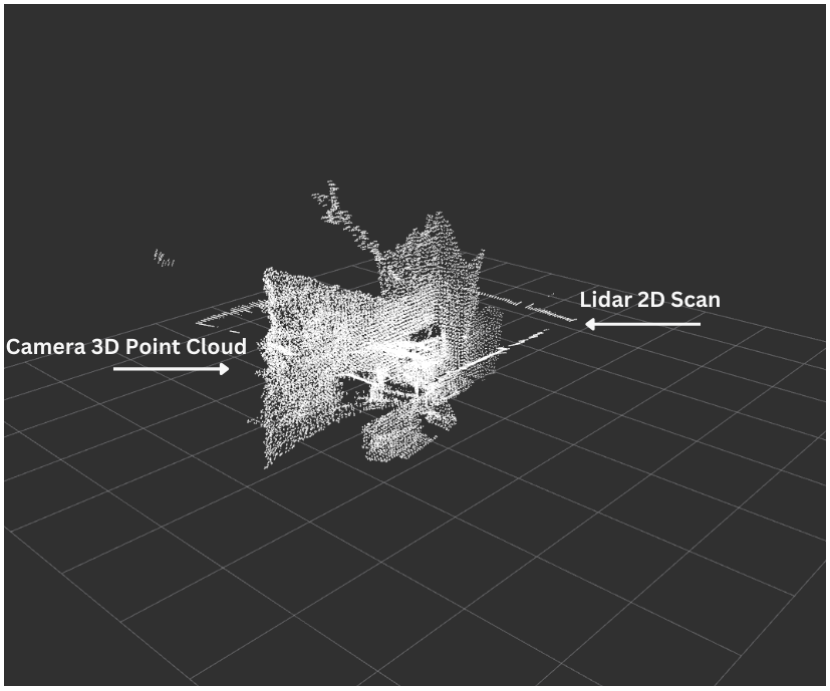


Figure 4.8: 3D environmental map generated from fused LiDAR and RealSense data

9. Development of 3D Localization Algorithms

With the fused sensor data available, we began implementing 3D localization techniques. These allow the robot to estimate its full 6-DOF (degrees of freedom) pose within the environment. This process leverages a combination of SLAM algorithms, point cloud matching, and Kalman filtering for sensor fusion. In particular, the use of an Extended Kalman Filter (EKF) allows us to combine uncertain measurements from different sensors to improve localization accuracy.

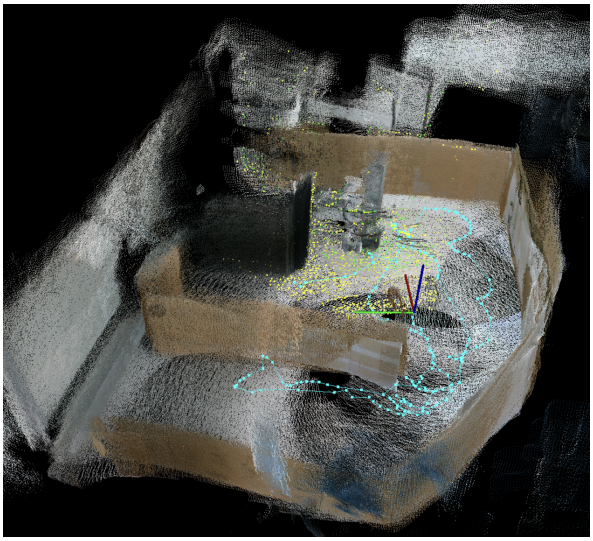


Figure 4.9: Robot localization in a 3D mapped environment

10. Implementation of Path Planning Using A* Algorithm

For autonomous navigation, we implemented the A* search algorithm to find optimal paths in the robot's environment. A* uses a heuristic-based cost function defined as:

$$f(n) = g(n) + h(n)$$

Where:

- $g(n)$ is the cost from the starting node to the current node.
- $h(n)$ is the estimated cost from the current node to the goal (Euclidean distance).

In 3D space, the heuristic function becomes:

$$h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2 + (z_{goal} - z_n)^2}$$

11. Modified A* Algorithm for Smoother Paths

To improve the quality of generated paths, we modified the A* algorithm by introducing a distance-based cost term that encourages smooth transitions between nodes:

$$f(n) = g(n) + h(n) + \alpha d(n)$$

Where:

- $d(n)$ is the Euclidean distance from the previous node.
- α is a tunable parameter that balances smoothness and optimality.

This modification helps reduce sharp turns and promotes trajectories that are more feasible for physical robot movement.

12. 3D Reconstruction and Multi-Floor Mapping

To visualize the results of our SLAM-based mapping, we performed a multi-floor mapping experiment in which the robot explored two separate floors of a building. The mapping was carried out consecutively, capturing LiDAR and depth data on each floor using the integrated sensor suite. After data collection, the point clouds representing each floor were exported and aligned manually. We used CloudCompare for point cloud cleaning and initial alignment through manual transformation and filtering operations. The combined point cloud was then imported into Blender, where fine-tuning of the alignment was done, and a mesh was generated to give a more structured representation of the environment.

Figure 4.10: Multi-floor 3D Mapping Point Clouds

The resulting point clouds were processed using CloudCompare and Blender to enhance their quality and prepare them for visualization. In CloudCompare, we applied various filtering techniques such as:

- **Statistical outlier removal** to eliminate noisy or floating points.
- **Smoothing filters** to reduce surface irregularities while preserving structure.
- **Simplification and decimation** to reduce the number of vertices for easier processing without significant loss of detail.

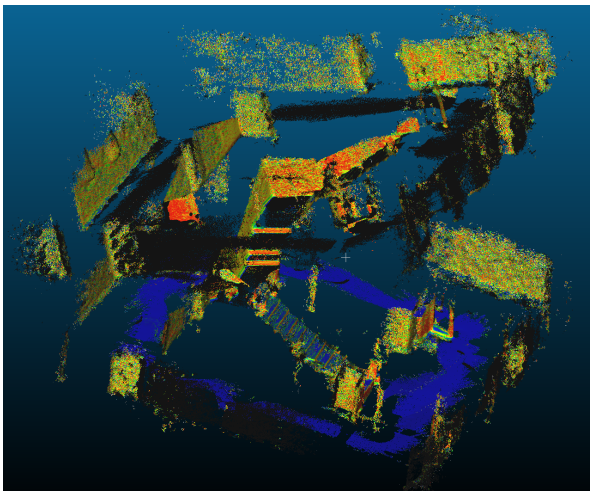


Figure 4.11: Filtered Point Clouds

After initial cleaning, the point clouds were imported into Blender, where advanced reconstruction techniques were used. Specifically, we utilized the **Geometry Nodes** workflow,

which allows for a non-destructive, procedural approach to converting point cloud data into mesh models. The pipeline consisted of the following steps:

1. **Point Cloud to Volume:** The imported point cloud was converted to a volumetric representation using a density-based volume conversion node.
2. **Volume to Mesh:** The volume was then converted into a mesh, enabling the creation of continuous surfaces from discrete point data.
3. **Post-Processing:** The resulting mesh was further refined through manual editing or additional modifiers to remove artifacts and improve aesthetics.

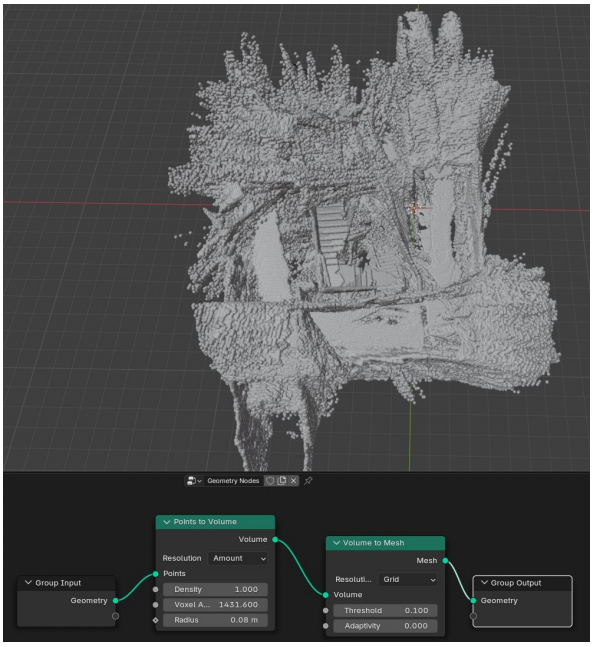


Figure 4.12: Point Clouds to Mesh

Chapter 5: **Challenges and Solutions**

- One of the main challenges we faced during the project was the compatibility of ROS packages with different versions of Ubuntu. Some packages we needed were only available for ROS Melodic, while others were designed for ROS Noetic, which led to difficulties in using them simultaneously. This version mismatch required us to find workarounds, as there was no straightforward solution to ensure compatibility across different systems.

  To address this, we manually worked on integrating the packages ourselves by modifying certain dependencies and adapting the packages to work with our existing ROS Melodic environment. This process involved careful version control and testing to ensure that the modified packages could run smoothly without introducing new issues. Going forward, we plan to continue refining these integrations and consider upgrading to a newer ROS version if it simplifies the process while maintaining system stability.

- Another significant challenge we faced was managing the computational load associated with sensor fusion and real-time SLAM processing. Integrating the depth camera and LiDAR data increased the demands on our system, particularly during the generation of fused 3D maps. This led to delays in mapping and localization, which affected robot performance in dynamic environments.

  To address this issue, we began transitioning our computational processes to the NVIDIA Jetson platform, leveraging its GPU capabilities to optimize real-time performance. The Jetson's parallel processing power allows us to handle the increased data throughput from multiple sensors, enabling smoother mapping and faster SLAM computations.

- Another challenge involved minor misalignments in the fused data due to static transformations applied during sensor synchronization. To improve alignment accuracy, we are now exploring dynamic calibration methods. These include using adaptive transformation matrices based on real-time feedback from the sensors to ensure more precise data fusion. These refinements will improve the overall reliability of the SLAM system, especially in complex environments.

- Another significant challenge was the quality of the raw point clouds. These were often noisy and required careful filtering and cleaning using CloudCompare before being suitable for further processing. Even after cleaning, rendering the resulting large mesh models in Blender was a demanding task. The high density of points and complexity of the generated models slowed down performance significantly and, in some cases, led to crashes during rendering or editing.

- Additionally, integrating the final mesh into simulation software presented further obstacles. When attempting to import the generated STL file into Gazebo, the file failed to load properly due to its large size and complexity. This prevented us from testing our robot within a fully simulated environment based on the mapped real-world scene, limiting our ability to evaluate the navigation algorithm in realistic conditions.

Through these efforts, we are addressing technical hurdles while enhancing the system's capability to achieve robust, multi-sensor SLAM.

Chapter 6: **Future Work**

---

1. Completion of Dynamic 2D Mapping and Wireless Navigation

- **Task:** Finalize SSH-based wireless control of the TurtleBot3 Burger and validate dynamic 2D mapping capability.
- **Timeline:** September 2024
- **Status:** Completed
- **Note:** Successfully mapped environment dynamically while operating wirelessly.

2. 2D Navigation and Autonomous Movement

- **Task:** Enable autonomous navigation using ROS navigation stack with 2D maps.
- **Timeline:** September–October 2024
- **Status:** Completed
- **Note:** Robot successfully navigates in the environment using the generated 2D dynamic maps.

3. 3D Mapping with Depth Camera Integration

- **Task:** Integrate depth camera to enable 3D mapping, test initial performance.
- **Timeline:** October 2024
- **Status:** Completed
- **Note:** Depth-based point cloud successfully used to generate 3D map.

4. Sensor Fusion and Kalman Filter Implementation

- **Task:** Fuse data from LiDAR, depth camera, and radar using Kalman filters.
- **Timeline:** October 15 – November 15, 2024
- **Status:** Completed
- **Note:** SLAM performance improved with sensor fusion.

5. 3D SLAM Development and Refinement

- **Task:** Implement full 3D SLAM system and begin tuning.
- **Timeline:** November 2024 – January 2025

- **Status:** Completed
- **Note:** 3D SLAM tested and refined for better accuracy and efficiency.

6. Jetson Nano Deployment and Gazebo Simulation

- **Task:** Transition SLAM to Jetson Nano and simulate mapping in Gazebo.
- **Timeline:** January – March 2025
- **Status:** Completed
- **Note:** System tested in Gazebo with TurtleBot3 in a 3D simulated world.

7. 2D Path Planning in 3D Environment

- **Task:** Implement 2D global/local path planning within a 3D map.
- **Timeline:** February 20 – March 15, 2025
- **Status:** Completed
- **Note:** Successfully planned and executed paths in 3D world using 2D navigation stack.

Future Improvements

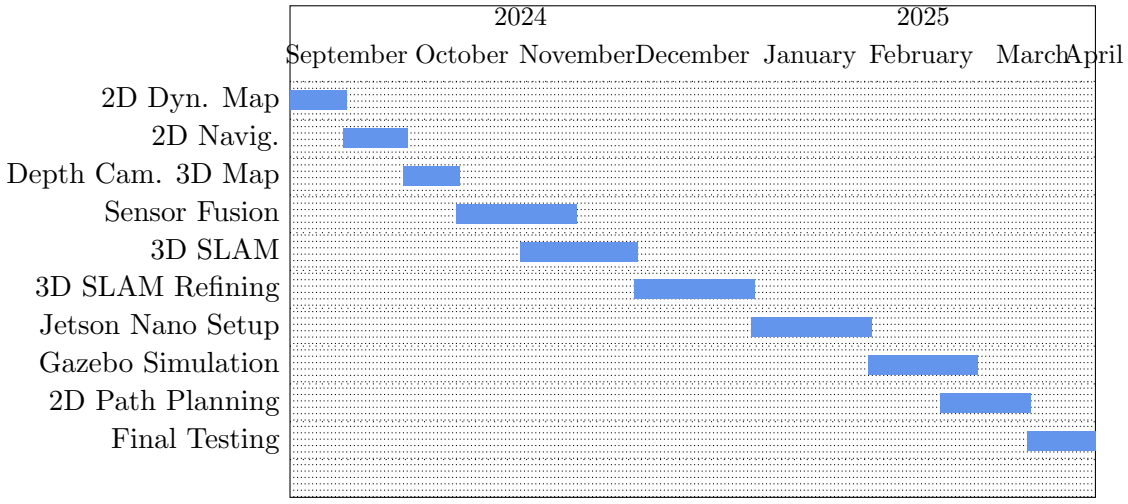
8. Elevation Mapping and 2.5D Path Planning

- **Task:** Upgrade SLAM system to support elevation maps and terrain-aware navigation.
- **Plan:** Implement elevation mapping (using OctoMap or Elevation Mapping packages) and plan paths over sloped or uneven terrain using 2.5D costmaps.

9. Extension to Larger Mobile Robot

- **Task:** Port SLAM and navigation system to a larger mobile platform for extended range and payload.
- **Plan:** Adapt SLAM, path planning, and sensor systems to work reliably on larger robots in more complex environments.

6.1 Gantt Chart



By systematically addressing these tasks, the project aims to deliver a scalable and high-performance SLAM solution, demonstrating its applicability in real-world robotic applications. Each phase contributes to refining the system's capabilities, ensuring a reliable and versatile platform for autonomous mobile robots.

Bibliography

[1] S. Gobhinath, K. Anandapoorani, K. Anitha, D. Dhivya Sri, and R. DivyaDharshini, "Simultaneous localization and mapping [slam] of robotic operating system for mobile robots," in *7th International Conference on Advanced Computing & Communication Systems (ICACCS)*, Department of EEE, Sri Krishna College of Engineering and Technology, Coimbatore-641008, 2021.

[2] R. S. Pol, V. N. Aher, S. V. Gaikwad, D. G. Bhalke, A. Y. Borkar, and M. T. Kolte, "Autonomous differential drive mobile robot navigation with slam, amcl using ros," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 5s, pp. 46–53, 2023.

[3] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 225–234, IEEE, 2007.

[5] J. Civera, A. J. Davison, and J. M. Martínez Montiel, "Inverse depth parametrization for monocular slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[6] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453)*, vol. 1, pp. 206–211, IEEE, 2003.

[7] M. Montemerlo and S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Springer Science & Business Media, 2007.

[8] L. Afonso, N. Lopes, A. J. Lima, D. Cruz, and P. Gaspar, "A slam method for the formula student driverless competition," 2021.

[9] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 380–407, 2004.

[10] V. S. Kalogeiton, K. Ioannidis, G. C. Sirakoulis, and E. B. Kosmatopoulos, "Real-time active slam and obstacle avoidance for an autonomous robot based on stereo vision," *Cybernetics and Systems*, vol. 50, no. 3, pp. 239–260, 2019.

[11] J. Nieto, J. Guivant, E. Nebot, and S. Thrun, "Real time data association for fastslam," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1, pp. 412–418, IEEE, 2003.

[12] J. J. Leonard and H. J. S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Robotics Research: The Ninth International Symposium*, pp. 169–176, Springer London, 2000.

[13] M. Jin, G. Zhou, Q. Wang, and H. Xiong, "Research and exploration of slam solutions based on mobile robots," in *International Conference on Intelligent Systems, Communications, and Computer Networks (ISCCN 2023)*, vol. 12702, pp. 600–605, SPIE, 2023.

[14] S. A. Li, Y. C. Chen, B. X. Wu, and H. M. Feng, "3d lidar slam-based systems in object detection and navigation applications," *Journal of the Chinese Institute of Engineers*, vol. 46, no. 8, pp. 912–925, 2023.

[15] H. Liu and J. Luo, "Yes-slam: Yolov7-enhanced-semantic visual slam for mobile robots in dynamic scenes," *Measurement Science and Technology*, vol. 35, no. 3, p. 035117, 2023.