

Faculty of Engineering.

Department of Control and Robotics Engineering.

Title

Semantic SLAM Mapping

Senior Project 2

Provided in fulfillment of the requirements for obtaining a bachelor's degree in Control and Robotics Engineering.

Prepared by

Suliman K. Al-Ouf

Supervised by

Dr. Sami Al-Issa

Eng. Marwa Al-Safadi

External Supervisor

Dr. Alaa Saadah

The University of Debrecen

Academic year

2024

Supervisor Certification

I certify that the preparation of this project entitled **Semantic SLAM Mapping**, Prepared by **Suliman k. Al-Ouf** was under my supervision.

Acknowledgment

I wish to express my sincere and profound gratitude to the individuals whose guidance, support, and inspiration have been instrumental in the completion of this graduation project and in shaping my academic and personal journey.

First and foremost, I am deeply indebted to my supervisors. To Dr. Sami Issa, for his unwavering encouragement and infectious optimism; he always greeted me with a smile and consistently pushed me to trust in my own abilities. My heartfelt thanks go to Dr. Alaa Saddah, who has been more than a supervisor, but a mentor, my first robotics coach, and a cherished friend. I am also immensely grateful to Dr. Marwa Al-Safadi for her incredible patience and meticulous guidance, especially in navigating the more than forty research papers that formed the foundation of this work.

I extend my special thanks to the professors at the university who have left an indelible mark on my education. To Dr. Rania Sandouk, the coach with whom we achieved the prestigious first-place victory in the All-Japan Sumo Robot Tournament. To Dr. Chadi Al-Bitar, who not only contributed significantly to this project but whose lectures I enjoyed immensely. My appreciation also goes to Dr. Ibrahim Chouaib, who served as my first judge and provided early, formative feedback, as well as to Dr. Talal Hammoud, Dr. Prof. Riad Al-Ghazzi, Eng. Maya, and Eng. Moafak for their invaluable contributions to my learning.

I owe a special debt of gratitude to the Dean of my College and my manager, Dr. Prof. Mouhib Al-Noukari. He saw a potential in me that I had not yet seen in myself, and his advice to change my major to Robotics and Control Engineering, though it extended my academic journey, set me on the path I was truly meant to follow. My sincere appreciation extends to all the professors, engineers, and management of the Syrian Private University (SPU) for fostering a challenging and supportive academic environment.

I must also acknowledge the Head of the Syrian Astronomical Association Muhammed AL-Assiry, whose passion for the cosmos first ignited my own passion to pursue a future in engineering.

Finally, this journey would not have been the same without my partners and best friends, the smartest and most passionate people I know: Eng. Adel SaadEddin and Eng. Mohammed Ali Mohammed Nabil Issa Al-Hammad (h4k0r). Your intellect, camaraderie (idk what it means but it sound right!), and shared enthusiasm were a constant source of motivation.

To all who have supported me, thank you.

Dedication

In loving memory of my grandfather. He would have been prouder than words could say, and he never missed a chance to show it. I know he would have loved to see Syria finally free.

To my father, the researcher and inventor Dr. Khaled, who set the bar of achievement so high and inspired me to reach for it.

To my mother, Burida Shammout, who has given me all the love and support anyone could ever need. Your belief in me is my foundation.

To my brother, Abd-Al-Rahman, and my sisters, Maria and Sophia.

To my uncles: Ammar, Abd-allah, who helped me find the courage to leave dentistry for engineering and Mohammad Shammout, Osama, Radwan. And to Bashar Al-Ouf, the most supportive person I know, who always has my back, no matter what.

To Teslars: Mahmoud, Tony, and Celine.

To my colleagues, the roboticists who shared this path with me: Yahea, Yamen, Hadi, and Lenar.

To Ayham, Mamdouh, and Bangin & Helen.

To my other family, the family I chose, the people I belong to: Sham my everything! Nadja, the bro. The blondie Nour. Hala. Hiba. Maher. Abd & Ram. Nagham. Giulianna. The old wise man, Luca. The little boy, Alejandro.

To all my friends, near and far.

And finally, to my cat, Bonita, for her unwavering commitment to waking me up at the crack of dawn—surely to ensure I was studying, and not at all because she was hungry.

Abstract

The rapid growth of autonomous service robots in sectors such as healthcare and hospitality necessitates robust, cost-effective navigation solutions. Current state-of-the-art SLAM systems often rely on expensive sensors and high-performance computing, creating a barrier to widespread adoption. This report addresses this gap by proposing a computationally lightweight, graph-based SLAM framework for resource-constrained platforms operating in large, multi-story environments. The primary contribution is the design of a framework that shows how two lightweight constraints — inter-floor structural alignment and an intra-floor soft Manhattan-World prior — could be combined to pursue global consistency in multi-story environments on affordable hardware. Building on soft Manhattan-World SLAM and prior inter-floor graph-constraint methods, the framework adapts these ideas to a low-cost, 2D-only sensor suite. The geometric map would be augmented with a lightweight, color-based semantic layer, enriching the map for human–robot interaction while avoiding the computational cost of deep object-detection models. The enabling robotic platform and data-registration pipeline were implemented and validated on a custom-built, low-cost robot, demonstrating real-time operation on a Raspberry Pi 5 (with the mapping loop estimated at roughly 10 ms of overhead). The proposed graph constraints are designed to reduce inter-floor and intra-floor drift; their quantitative validation against ground truth is identified as the primary direction for future work. This work establishes the viability of the foundational platform and proposes both a framework and an evaluation methodology for achieving robust, semantic, and globally consistent mapping on affordable hardware, paving the way for more capable and accessible service robots.

Table of figures

Figure	Caption	Page
Fig. 1.1	A graphical representation of the essential SLAM problem, showing the relationship between the robot's true (unknown) trajectory (x), its estimated trajectory (yellow triangles), landmarks (m), and the noisy observations (z) that connect them.	2
Fig. 1.2	A high-level block diagram of the proposed integrated SLAM framework, illustrating the primary data inputs, core processing modules, and the final map output.	4
Fig. 2.1	Typical particle distributions in filter-based SLAM. (a) In a feature-rich corridor, particle uncertainty is constrained. (b) At a dead end, uncertainty is low. (c) Raw odometry alone leads to a much less certain, more diffuse particle cloud.	6
Fig. 2.2	The practical effect of graph optimization on real-world datasets. The left column shows the initial, unoptimized trajectories and maps with significant drift, while the right column shows the globally consistent result after optimization. (a) 2D and 3D pose-graph datasets. (b) A large-scale bundle adjustment dataset.	7
Fig. 2.3	The common architecture of a modern SLAM system, divided into the front-end, which processes raw sensor data for local mapping, and the back-end, which performs global optimization and loop closure.	8
Fig. 2.4	Examples of early semantic mapping. (a) Object instances (e.g., robosapien, basketball) are recognized and their locations are annotated on a 2D metric map. (b) A floor plan showing a typical experimental environment with semantically labeled rooms and furniture.	9
Fig. 2.5	A dense mapping system using planar surfaces as semantic primitives instead of points. (a) The factor graph formulation. (b-c) The resulting dense, structured 3D map.	10
Fig. 2.6	Example predictions from SegNet, a deep convolutional neural network for semantic segmentation. The network assigns a class label (e.g., road, building, car, pedestrian) to every pixel in an image.	11
Fig. 2.7	Examples of different semantic layers applied to 2D maps. (a) Obstacles with door detection. (b) Room-level classification (e.g., office, corridor). (c) Instance-level object detection, showing the location of beds, couches, and sinks.	12
Fig. 2.8	An illustration of map maintenance for dynamic environments. As new observations are made, the age of cell data is updated (represented by gray levels), allowing the system to distinguish static structure from transient objects.	13
Fig. 2.9	An example of aligning multi-story building maps using structural priors. (a) Individual floor graphs are misaligned before optimization but are correctly merged by a global constraint. (b) The resulting alignment of the 2D occupancy maps from two different floors.	15
Fig. 2.10	Real-time dense 3D mapping with Kintinuous: (a) A dense volumetric representation (TSDF) is used locally and shifted along with the camera. The pose graph is drawn in pink with small cuboids as camera poses that have cloud slices associated with them. Loop closure on the pose graph is transferred to the point cloud via mesh deformation. (b) Sequence over two floors of an apartment with over six million vertices that was constructed including online loop closure. Small details such as bathroom fixtures and objects around the environment are clearly reconstructed. [9]	14
Fig. 2.11	Visualization tools for SLAM evaluation. (a) A plot of the absolute trajectory error (ATE) over a path, provided by tools like the evo package. (b-c) Screenshots from SLAMBench, a framework for holistic, multi-objective benchmarking of SLAM systems.	17

Fig. 2.12	An example of a ground truth point cloud from the EuRoC MAV dataset, recorded with a high-precision motion capture system. Such datasets are essential for the quantitative evaluation of SLAM algorithm accuracy.	18
Fig. 3.1	The commercial hoverboard system adapted for the robot's actuation, showing the internal layout of the battery, main circuit, and wheel motors.	22
Fig. 3.2	The custom-built, low-cost robotic platform used for all experimental validation in this research.	23
Fig. 3.3	Technical specifications and dimensions of the 350W brushless DC (BLDC) hub motors repurposed from the hoverboard.	24
Fig. 3.4	Pinout diagram of the repurposed hoverboard mainboard, which serves as the dual-motor driver. The availability of such open-source, reverse-engineered diagrams enables the use of custom firmware.	24
Fig. 3.5	The Raspberry Pi 5 single-board computer, which serves as the robot's central processing unit for all high-level computations.	25
Fig. 3.6	The 3irobotix Delta-2A 2D LiDAR, the primary sensor for geometric mapping and localization.	25
Fig. 3.7	The two low-cost color cameras, mounted back-to-back (≈180° apart) on the platform, used to provide appearance-based color data for the lightweight semantic mapping layer.	26
Fig. 4.1	Examples of large-scale mapping outputs that require graph-based optimization to ensure global consistency. (Top) A map of the Stanford University campus. (Bottom) A 3D map of the Gates Computer Science building.	30
Fig. 4.2	Extraction of dominant line features from environmental point clouds. Such planar and linear structures are used as robust geometric signatures for enforcing both inter-floor and intra-floor (Manhattan) constraints.	32
Fig. 4.3	An illustration of SoMaSLAM, a system that applies soft Manhattan-World constraints. The method successfully maps both orthogonal "Manhattan-World" areas and more complex "Non-Manhattan" sections of the environment.	34
Fig. 5.1	RViz Visualization of Stacked Semantic Maps (This figure shows a screenshot from RViz. Multiple 2D occupancy grid maps of the same room are displayed, stacked on top of each other with a vertical offset. Obstacles in the maps are colored (e.g., blue for walls, brown for a door), demonstrating the fusion of geometric data from the LiDAR and semantic color data from the camera.)	39

Table of Abbreviations

Abbreviation	Full Form
ATE	Absolute Trajectory Error
BGR	Blue, Green, Red
BLDC	Brushless Direct Current (Motor)
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FOC	Field-Oriented Control
g2o	General Graph Optimization
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LTS	Long-Term Support
MW	Manhattan-World
OCR	Optical Character Recognition
PF	Particle Filter
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RBPF	Rao-Blackwellized Particle Filter
RGB-D	Red Green Blue - Depth
ROS	Robot Operating System
RPE	Relative Pose Error
RViz	ROS Visualization
SLAM	Simultaneous Localization and Mapping
TSDF	Truncated Signed Distance Function
USB	Universal Serial Bus
USART	Universal Synchronous/Asynchronous Receiver/ Transmitter
YOLO	You Only Look Once

Chapter 1
Introduction

1.1. The Rise of Autonomous Service Robots

The 21st century has witnessed the rapid integration of robotics into various facets of daily life, extending far beyond traditional industrial automation. A significant driver of this evolution is the service robotics sector, where autonomous systems are increasingly deployed in public and commercial spaces to assist humans and streamline operations. In healthcare, service robots are being utilized for tasks such as medication and meal delivery, reducing the burden on nursing staff and minimizing human contact [1]. Similarly, the hospitality industry has begun to adopt robots to handle guest inquiries and provide concierge services, aiming to enhance guest experiences and improve operational efficiency [2]. The successful deployment of these robots is fundamentally reliant on their ability to navigate their environments safely, efficiently, and intelligently.

1.2. The Foundational Role of SLAM

For an autonomous mobile robot to perform any useful task, it must possess the ability to create a map of its environment and simultaneously determine its own location within that map. This fundamental challenge is known as Simultaneous Localization and Mapping (SLAM), a concept visualized in Fig. 1.1 [3]. SLAM is the foundational technology that enables autonomous navigation in unknown environments where external positioning systems like GPS are unavailable or unreliable. A robust SLAM system empowers a robot to understand its surroundings, plan paths, and execute tasks with precision and reliability. Over the past two decades, SLAM has evolved from a theoretical problem to a robust and mature field of study, forming the backbone of modern autonomous systems from domestic vacuum cleaners to planetary rovers [4].

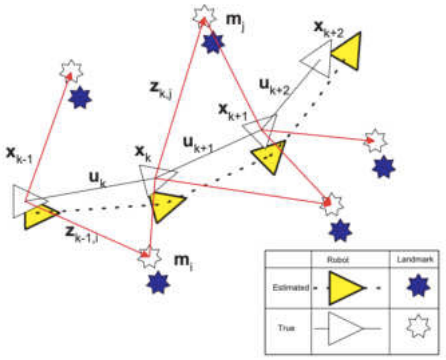


Fig. 1.1: A graphical representation of the essential SLAM problem, showing the relationship between the robot's true (unknown) trajectory (x), its estimated trajectory (yellow triangles), landmarks (m), and the noisy observations (z) that connect them. [3]

1.3. The Research Gap: Practical Autonomy on a Budget

Despite significant advances, a persistent gap exists between the cutting-edge SLAM research demonstrated in academic labs and the practical requirements of commercial service robots. State-of-the-art academic systems often achieve remarkable accuracy by leveraging expensive sensors, such as 3D LiDAR scanners, and high-performance

GPUs for complex computations [5]. This approach, while valuable for research, results in systems that are too costly and power-intensive for widespread commercial deployment. Consequently, a critical need has emerged for SLAM solutions that can deliver "good enough" performance on affordable, resource-constrained hardware. This project directly addresses this need by focusing on achieving robust, globally consistent mapping in a challenging, real-world environment using a low-cost sensor suite.

1.4. Research Objectives

The primary objective of this research is to design a novel SLAM framework capable of generating a globally consistent, semantically enriched 2D map of a large, multi-story indoor environment using a low-cost robotic platform, and to implement and validate the foundational components of this system.

This primary objective is supported by the following secondary objectives:

1. To design a graph-based SLAM system that enforces global, inter-floor alignment by leveraging computationally inexpensive architectural priors, specifically the structural similarity between floors.

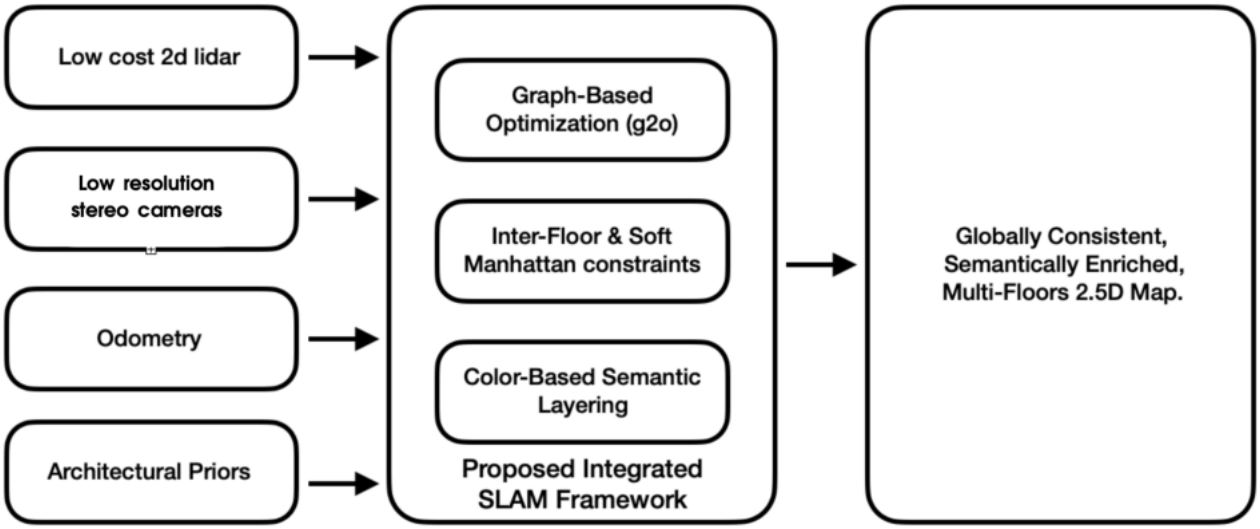
2. To propose a lightweight semantic mapping layer using color data from a low-cost camera to enrich the geometric map without relying on computationally expensive deep learning models.

3. To build and validate a low-cost robotic platform capable of running the entire proposed software stack in real-time.

1.5. Contribution and Report Structure

The contribution of this report is threefold. First, it presents a system-level design that adapts and combines two known lightweight techniques — inter-floor graph constraints in the spirit of Karg et al. [20] and a soft Manhattan-World prior in the spirit of SoMaSLAM [41] — for a low-cost, 2D-only platform, and integrates them with a lightweight color-based semantic layer within a single graph-optimization formulation (summarized in Fig. 1.2). The aim is not to claim these constraints as individually new, but to show how their combination could enable globally consistent, multi-story mapping on hardware where state-of-the-art methods are impractical. Second, this work presents the detailed specification and open-source implementation of a fully integrated, low-cost robotic platform, validated as a real-

time testbed and offered as an accessible blueprint for SLAM research on a constrained budget. Third, it identifies a concrete gap in the available evaluation resources — the absence of a public benchmark for multi-story indoor mapping captured with low-cost 2D LiDAR and camera sensors — and proposes such a dataset (MSSR-25) as a direction for future work. The quantitative validation of the proposed constraints is, by design, left to that future work.



The remainder of this report is structured as follows. Chapter 2 provides a critical review of the existing literature in the SLAM field. Chapter 3 details the robotic platform's design and configuration. Chapter 4 describes the proposed SLAM framework methodology. Chapter 5 presents the experimental setup and evaluation results. Finally, Chapter 6 concludes the report.

Chapter 2
Literature Review

2.1. SLAM Fundamentals

The SLAM problem is often described as a "chicken-and-egg" problem: a good map is required to accurately determine location (localization), while an accurate location is required to build a good map (mapping). Modern SLAM algorithms solve this duality by treating it as a joint probabilistic estimation problem [6].

2.1.1. Probabilistic SLAM

The dominant paradigm in SLAM is probabilistic, which explicitly models the uncertainty inherent in sensor measurements and robot motion. These methods can be broadly categorized into two main families: filter-based approaches and graph-based approaches [4].

2.1.2. Filter-Based SLAM

Filter-based methods, such as those using the Extended Kalman Filter (EKF) or Particle Filters (PF), sequentially estimate the robot's current pose and the map. Among the most successful implementations of filter-based SLAM is the Gmapping algorithm, which leverages a Rao-Blackwellized Particle Filter (RBPF) to efficiently generate 2D occupancy grid maps. As detailed by Grisetti, Stachniss, and Burgard, the key innovation of Gmapping lies in its adaptive resampling technique and improved proposal distribution, which mitigates particle impoverishment and allows for highly accurate mapping in real-time within structured environments, as the particle distributions in Fig. 2.1 illustrate [7].

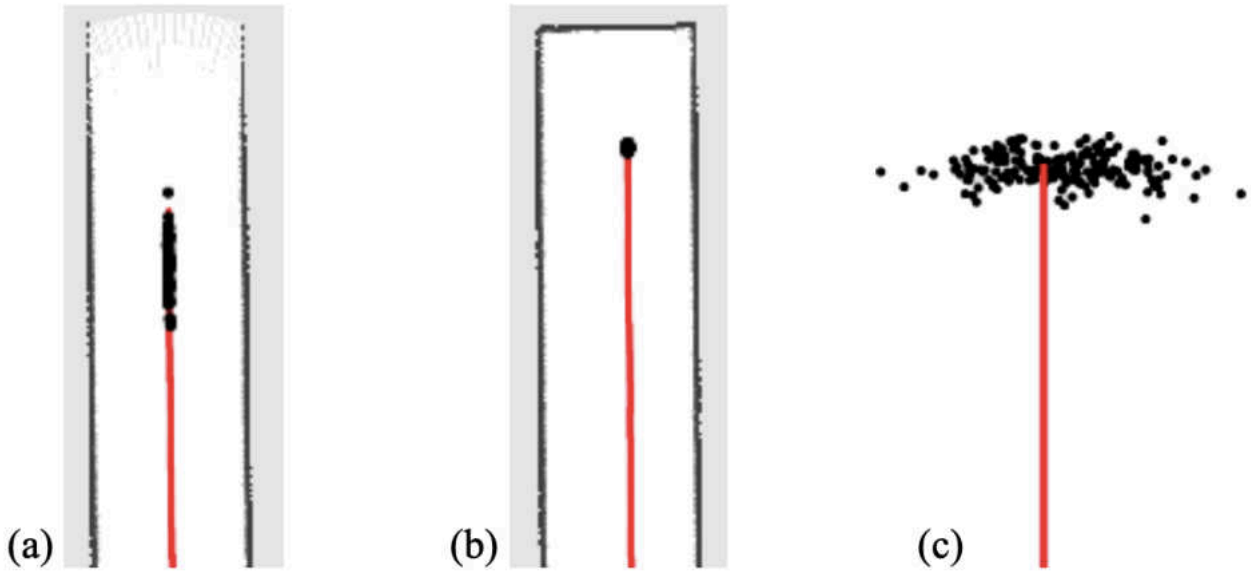


Fig. 2.1: Typical particle distributions in filter-based SLAM. (a) In a feature-rich corridor, particle uncertainty is constrained. (b) At a dead end, uncertainty is low. (c) Raw odometry alone leads to a much less certain, more diffuse particle cloud. [7]

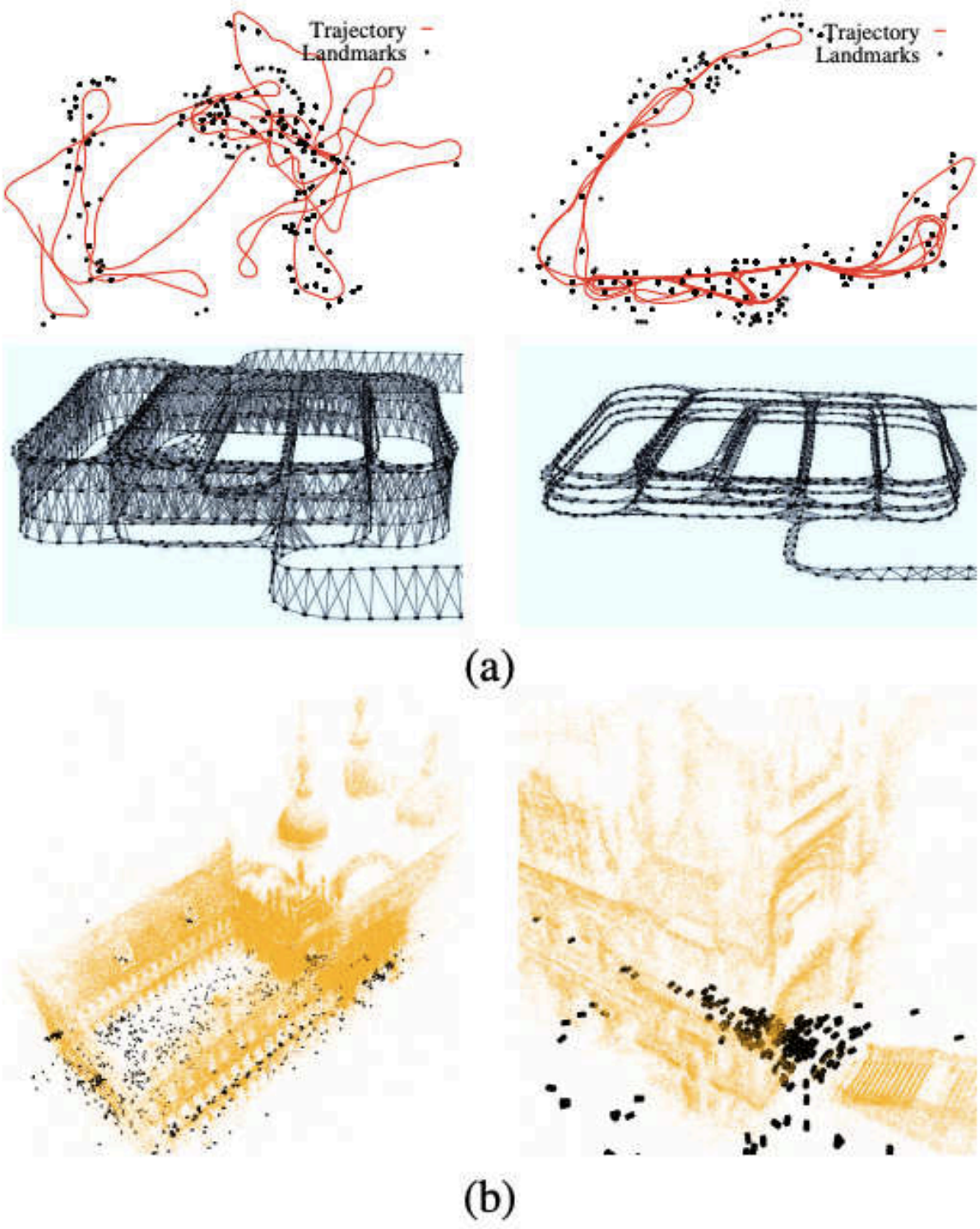


Fig. 2.2: The practical effect of graph optimization on real-world datasets. The left column shows the initial, unoptimized trajectories and maps with significant drift, while the right column shows the globally consistent result after optimization. (a) 2D and 3D pose-graph datasets. (b) A large-scale bundle adjustment dataset. [8]

However, the very nature of this filter-based approach, which estimates the *current* pose based on the immediate past, makes it inherently vulnerable to long-term drift. While effective for single-floor deployments, Gmapping lacks a native mechanism for large-scale loop closure. While its particle filter provides limited implicit

correction when revisiting an area, it has no *explicit* loop-closure or global pose-graph optimization step, and this implicit correction degrades sharply with low-cost, low-resolution LiDAR — so it cannot globally redistribute accumulated error across a large or multi-story map. Consequently, when applied to large or multi-story environments, accumulated odometric errors become very difficult to correct after the fact, leading to severe map distortions—such as misaligned or rotated floors—that motivate this research. The inability of even a state-of-the-art particle filter like Gmapping to ensure global consistency on its own forms the central justification for adopting a graph-based optimization framework. Such frameworks, like g2o [8], are explicitly designed to incorporate non-sequential, global constraints and find a globally optimal solution by minimizing the error over the entire robot trajectory, a process whose dramatic effect is visualized in Fig. 2.2.

2.1.3. Graph-Based SLAM

Graph-based SLAM has become the state-of-the-art approach for generating large-scale, globally consistent maps [8]. In this formulation, which is typically structured into a front-end and a back-end (Fig. 2.3), the problem is modeled as a graph where nodes represent robot poses at different points in time and sensor-derived landmarks. Edges between nodes represent spatial constraints derived from sensor measurements (e.g., odometry) or landmark observations. When a robot re-observes a previously visited location (a "loop closure"), a new edge is added to the graph, linking two non-sequential nodes. The entire graph can then be passed to a nonlinear least-squares optimizer, which adjusts all node positions to find the most probable map and trajectory given all constraints [9]. This ability to incorporate global constraints makes graph-based SLAM inherently better suited for large-scale and multi-session mapping, forming the core of our proposed system.

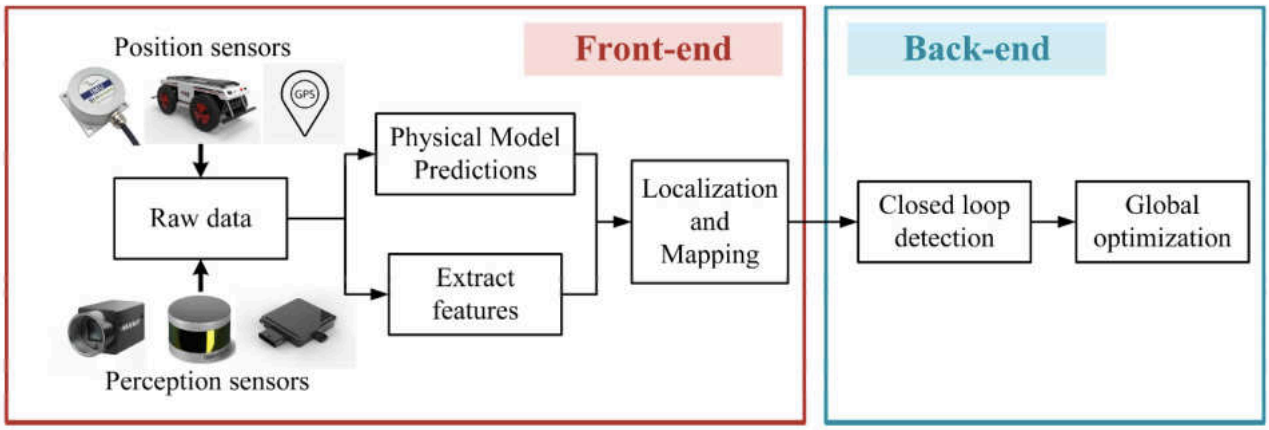


Fig. 2.3: The common architecture of a modern SLAM system, divided into the front-end, which processes raw sensor data for local mapping, and the back-end, which performs global optimization and loop closure. [5]

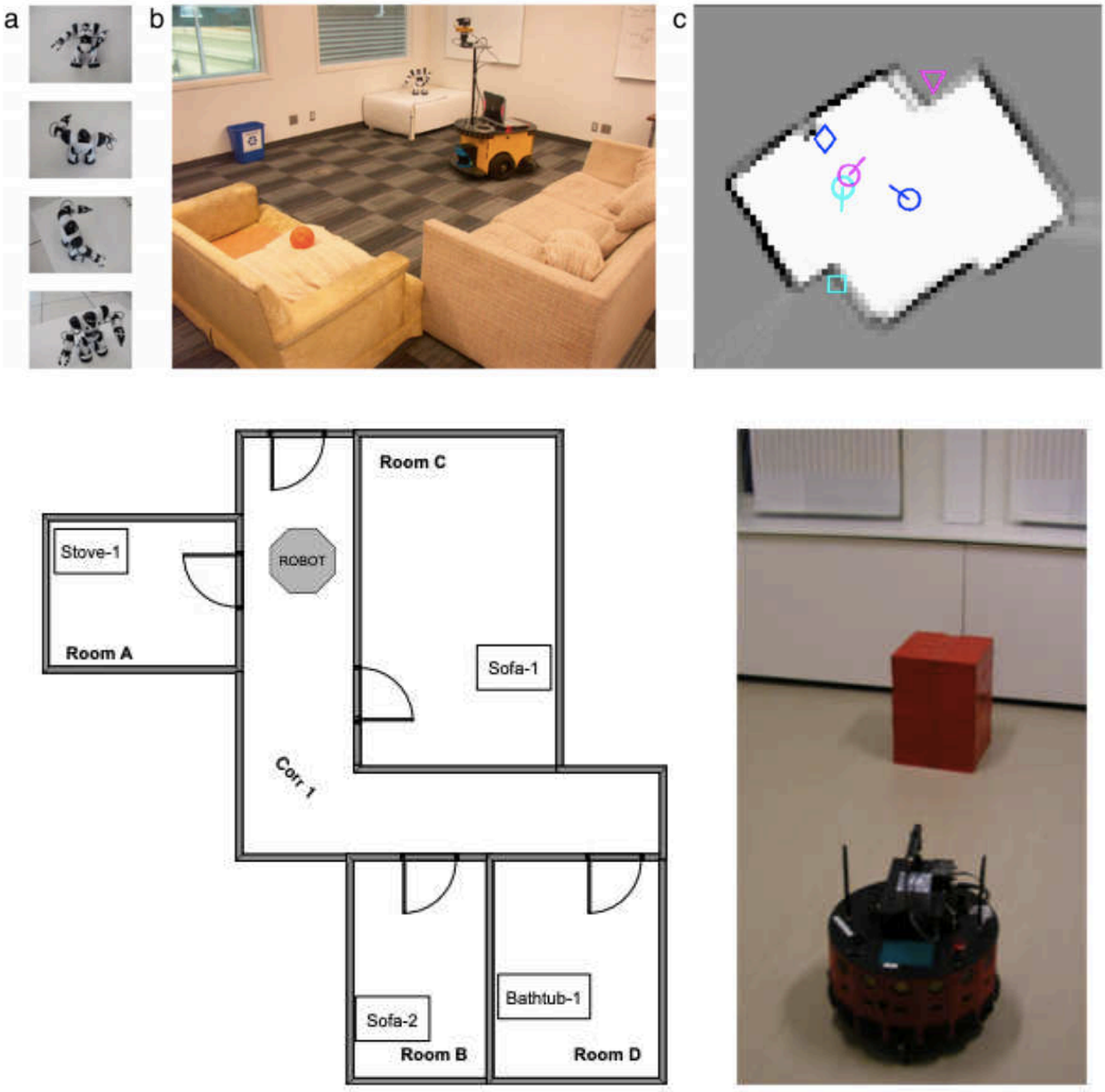
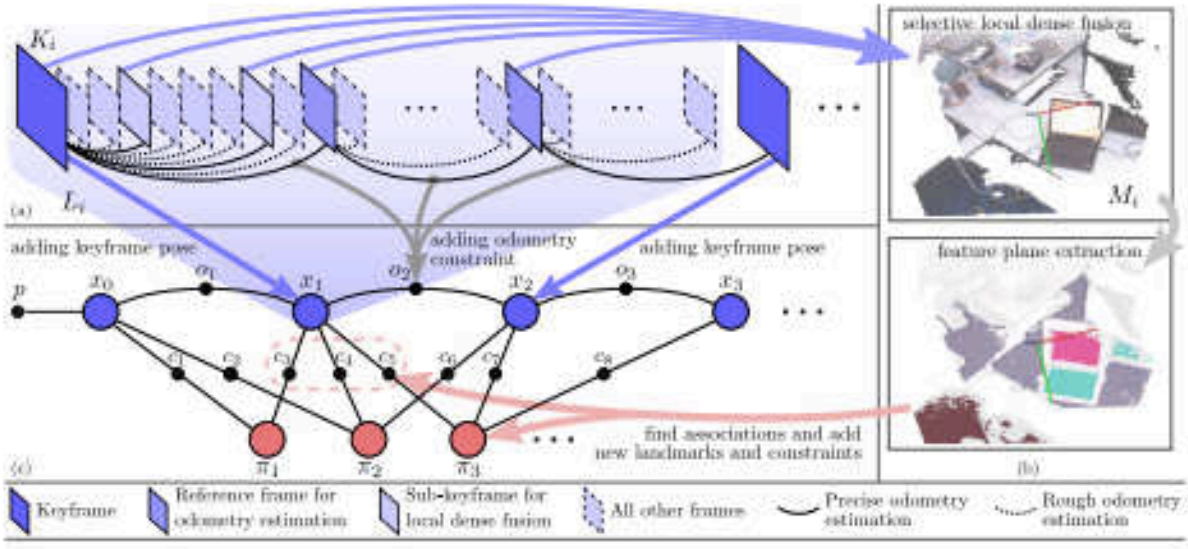


Fig. 2.4: Examples of early semantic mapping. (a) Object instances (e.g., robosapien, basketball) are recognized and their locations are annotated on a 2D metric map. [10] (b) A floor plan showing a typical experimental environment with semantically labeled rooms and furniture. [11]

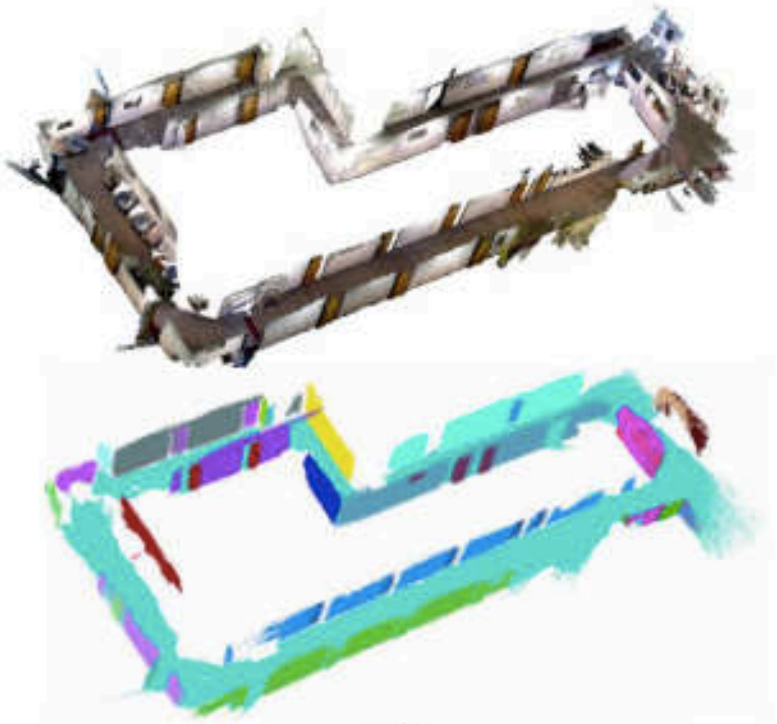
2.2. Semantic SLAM

Traditional SLAM produces a geometric map, which is useful for navigation but lacks any understanding of the objects or spaces it represents. Semantic SLAM aims to bridge this gap by creating maps that are not only geometrically accurate but also enriched with semantic meaning [10]. This allows for more natural human-robot interaction (e.g., "go to the kitchen") and more intelligent robot behavior.

Early semantic mapping approaches often relied on detecting specific objects like doors or furniture to serve as landmarks (Fig. 2.4) [11], while other methods use geometric primitives such as planes (Fig. 2.5). More recent approaches leverage the



(a)



(b)



(c)

Fig. 2.5: A dense mapping system using planar surfaces as semantic primitives instead of points. (a) The factor graph formulation. (b-c) The resulting dense, structured 3D map. [9]

power of deep learning, using Convolutional Neural Networks (CNNs) for object detection [12] or semantic segmentation (illustrated in Fig. 2.6) [13] to build dense, object-level maps. For example, systems can use models like YOLO or Mask R-CNN to identify instances of objects and integrate them into a map, leading to various types of semantic annotations as shown in Fig. 2.7 [14]. While powerful, these methods typically require significant computational resources (i.e., GPUs), making them unsuitable for our target low-cost platform. This computational barrier necessitates the exploration of alternative, lightweight approaches, such as the appearance-based method proposed in this report.

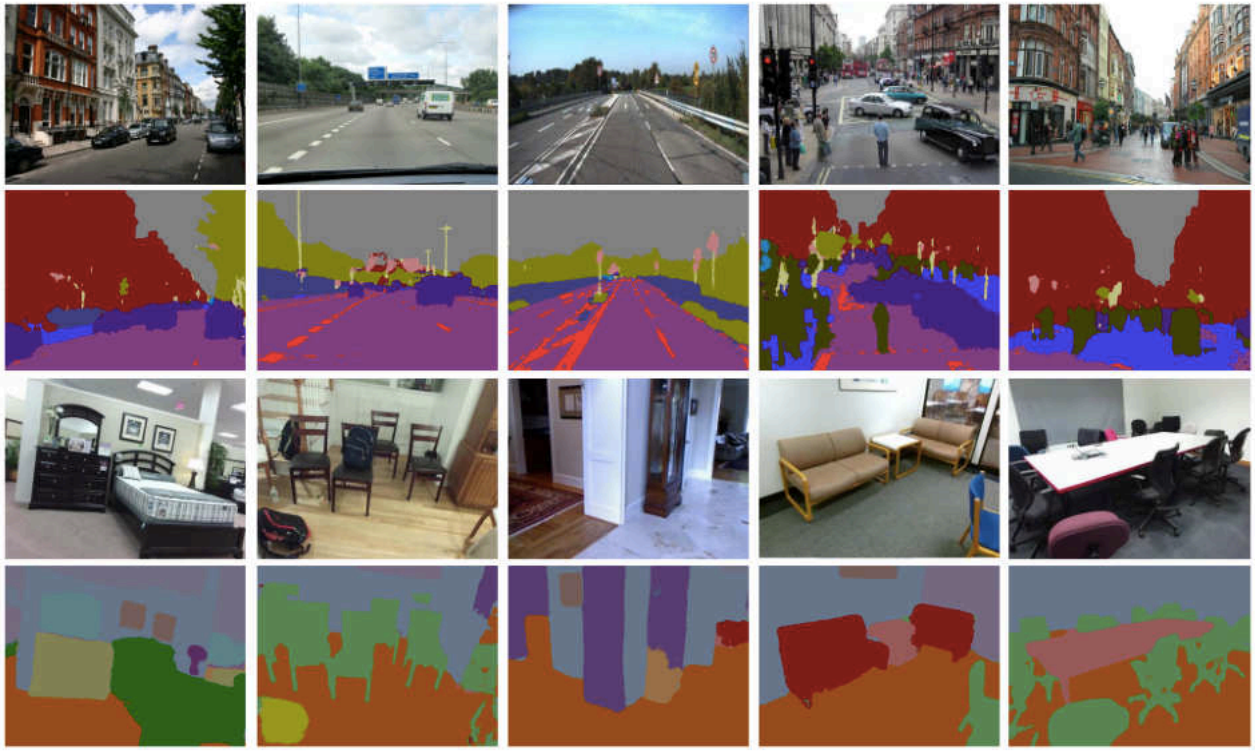


Fig. 2.6: Example predictions from SegNet, a deep convolutional neural network for semantic segmentation. The network assigns a class label (e.g., road, building, car, pedestrian) to every pixel in an image. [13]

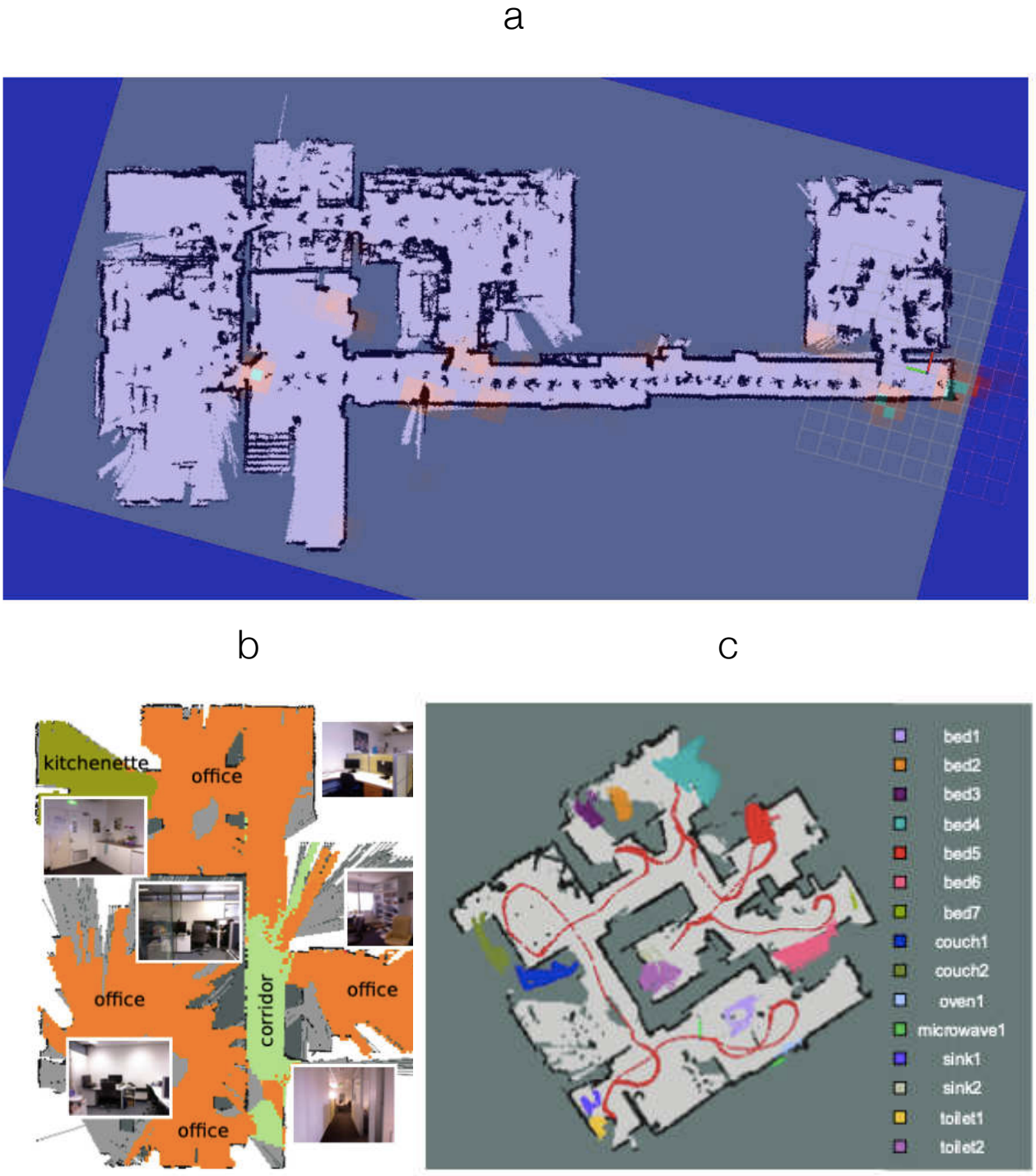


Fig. 2.7: Examples of different semantic layers applied to 2D maps. (a) Obstacles with door detection. (b) Room-level classification (e.g., office, corridor). (c) Instance-level object detection, showing the location of beds, couches, and sinks. [14]

2.3. SLAM in Dynamic and Multi-Story Environments

2.3.1. Dynamic Environments

Most SLAM algorithms assume a static world. This assumption is often violated in real-world environments populated by humans and moving objects. Robust SLAM systems must be able to differentiate between static and dynamic parts of the environment to avoid corrupting the map [15]. This can be achieved by tracking

moving objects [16] or by developing map maintenance strategies that can update or remove parts of the map that have changed over time (see Fig. 2.8) [17].

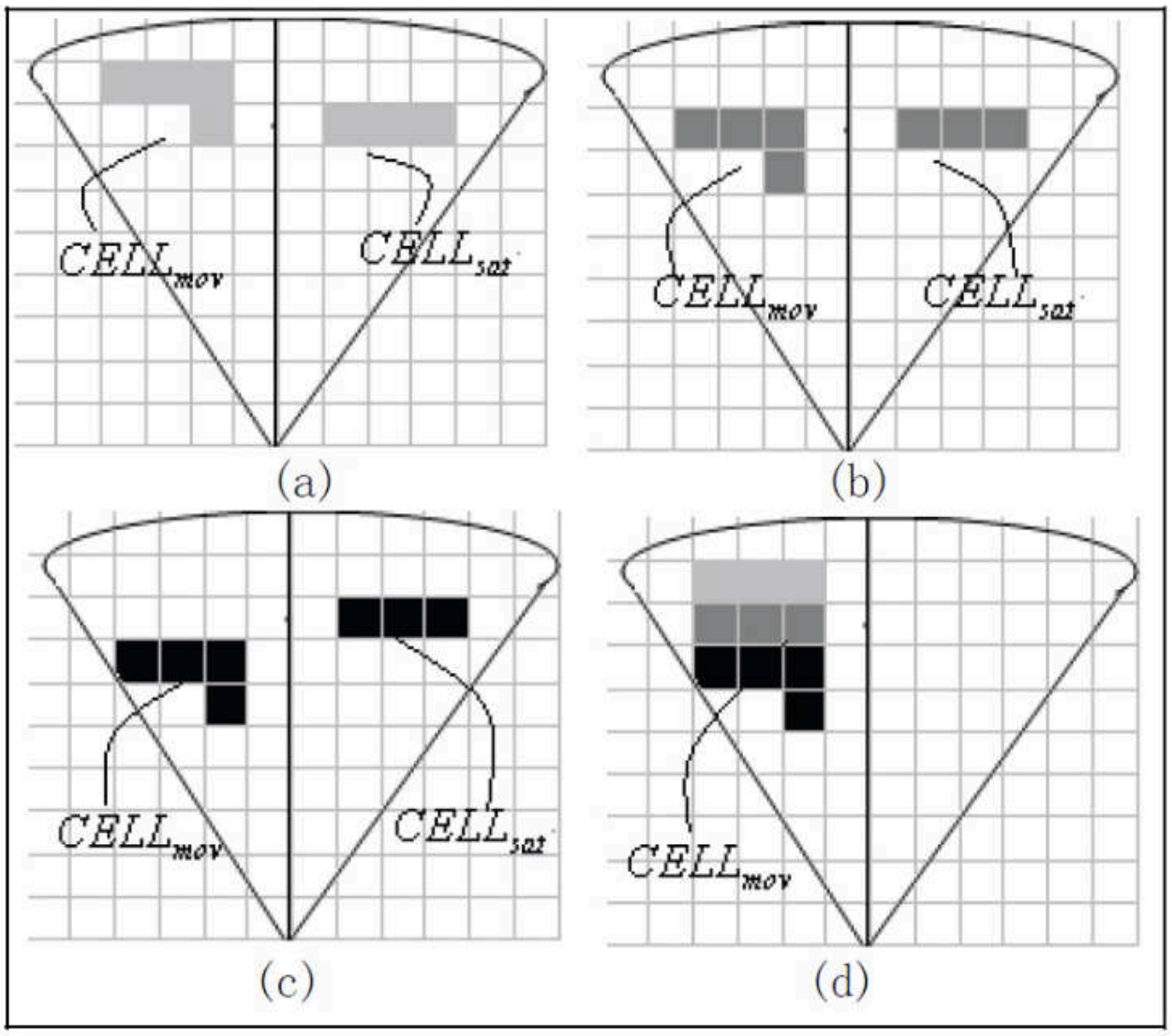


Fig. 2.8: An illustration of map maintenance for dynamic environments. As new observations are made, the age of cell data is updated (represented by gray levels), allowing the system to distinguish static structure from transient objects. [17]

2.3.2. Multi-Story Environments

Mapping multi-story buildings presents a unique challenge. While individual floors can be mapped using standard 2D SLAM, there is no straightforward way to align these maps into a globally consistent 3D structure using only 2D sensors, though dense reconstructions have been achieved with 3D sensors as shown in Fig. 2.9. Approaches to solve this often involve detecting key vertical structures like elevators or stairwells [18] or using visual place recognition to find inter-floor correspondences [19]. Our approach falls into a category that exploits architectural priors to find this alignment. This concept, illustrated in Fig. 2.10, has been explored using methods such as Monte-Carlo Localization to align one floor's scan data against another's map [20]. However, this project proposes a more computationally efficient feature-based map matching technique suitable for our resource-constrained platform.

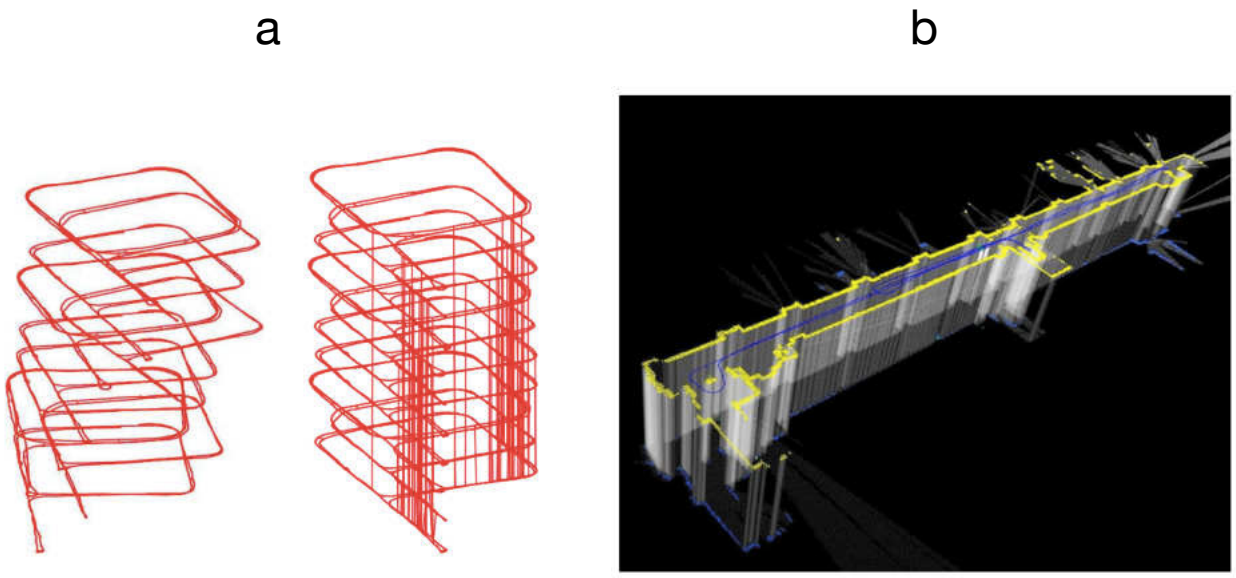


Fig. 2.10: An example of aligning multi-story building maps using structural priors. (a) Individual floor graphs are misaligned before optimization but are correctly merged by a global constraint. (b) The resulting alignment of the 2D occupancy maps from two different floors. [20]

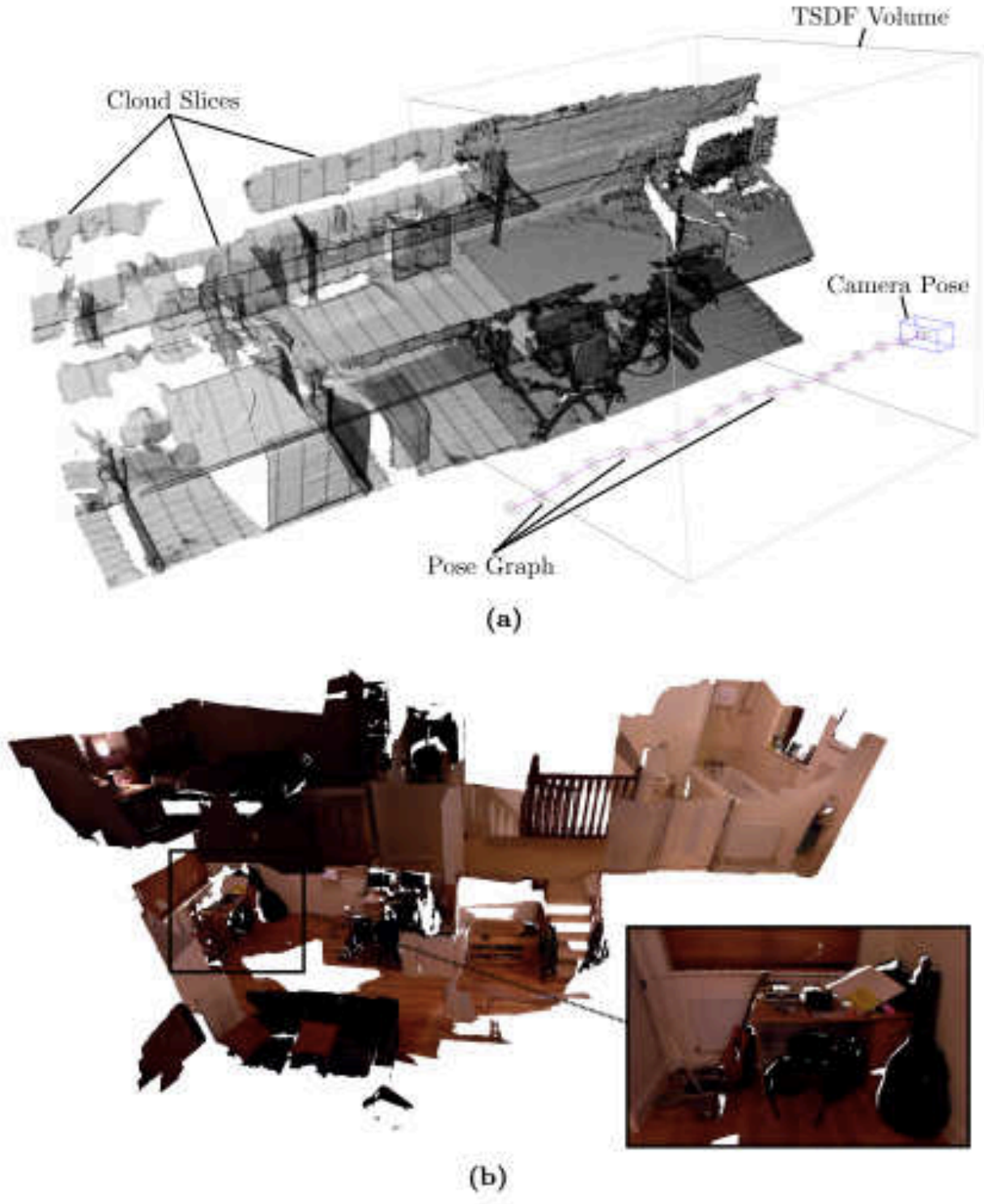


Fig. 2.9: Real-time dense 3D mapping with Kintinuous: (a) A dense volumetric representation (TSDF) is used locally and shifted along with the camera. The pose graph is drawn in pink with small cuboids as camera poses that have cloud slices associated with them. Loop closure on the pose graph is transferred to the point cloud via mesh deformation. (b) Sequence over two floors of an apartment with over six million vertices that was constructed including online loop closure. Small details such as bathroom fixtures and objects around the environment are clearly reconstructed. [9]

2.4. SLAM Evaluation Resources

2.4.1. Evaluation Tools

The robotics community has developed standardized tools for the objective evaluation of SLAM system performance. One of the most widely used is the `evo` Python package, which provides tools for analyzing the accuracy of estimated trajectories by comparing them against a ground truth [21]. It implements standard metrics such as Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). Benchmarking frameworks like SLAMBench [22] and RobotPerf [23] provide more comprehensive, multi-objective evaluation platforms, with outputs similar to those shown in Fig. 2.11, A comparison of these primary tools is summarized in Table 2.1.

Framework / Tool	Primary Function	Key Metrics	Relevance to this Report
evo	Trajectory Accuracy	ATE, RPE, path plots	Quantifies geometric accuracy.
SLAMBench	Holistic Performance	Accuracy, CPU, Memory, Power	Validates computational efficiency.
RobotPerf	System-level Throughput	Latency, data throughput	Assesses real-time software performance.

TABLE 2.1: Comparison of Common SLAM Evaluation Frameworks.

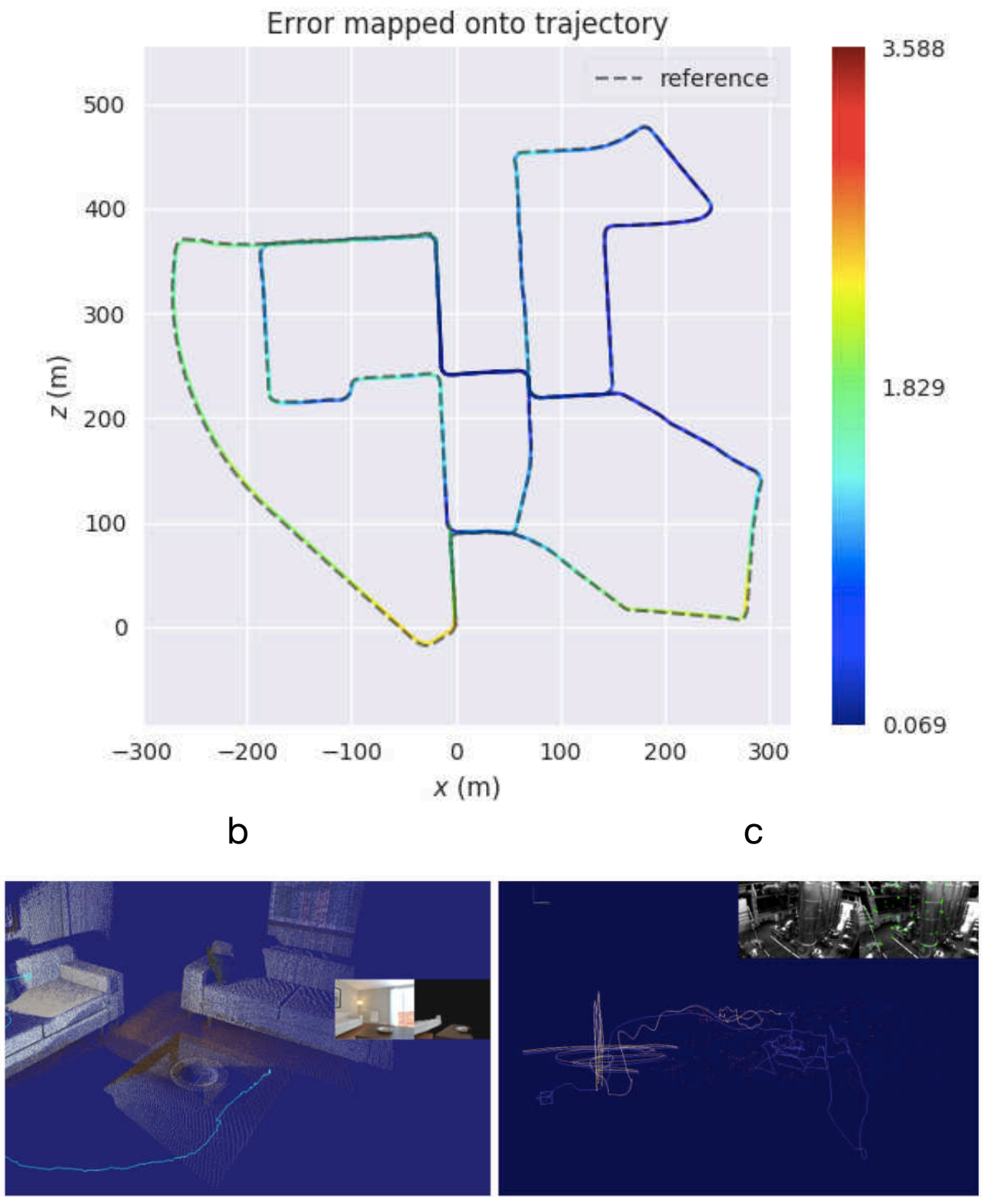


Fig. 2.11: Visualization tools for SLAM evaluation. (a) A plot of the absolute trajectory error (ATE) over a path, provided by tools like the `evo` package. [21] (b-c) Screenshots from SLAMBench, a framework for holistic, multi-objective benchmarking of SLAM systems. [22]

2.4.2. Public Datasets

The availability of public datasets has been a major catalyst for progress in SLAM research. Seminal datasets like the TUM RGB-D Benchmark [24] and the KITTI Vision Benchmark Suite [25] established rigorous evaluation standards for visual and LiDAR odometry. Other influential datasets include the EuRoC MAV dataset [26], which provides high-precision ground truth data (Fig. 2.12) for visual-inertial SLAM,

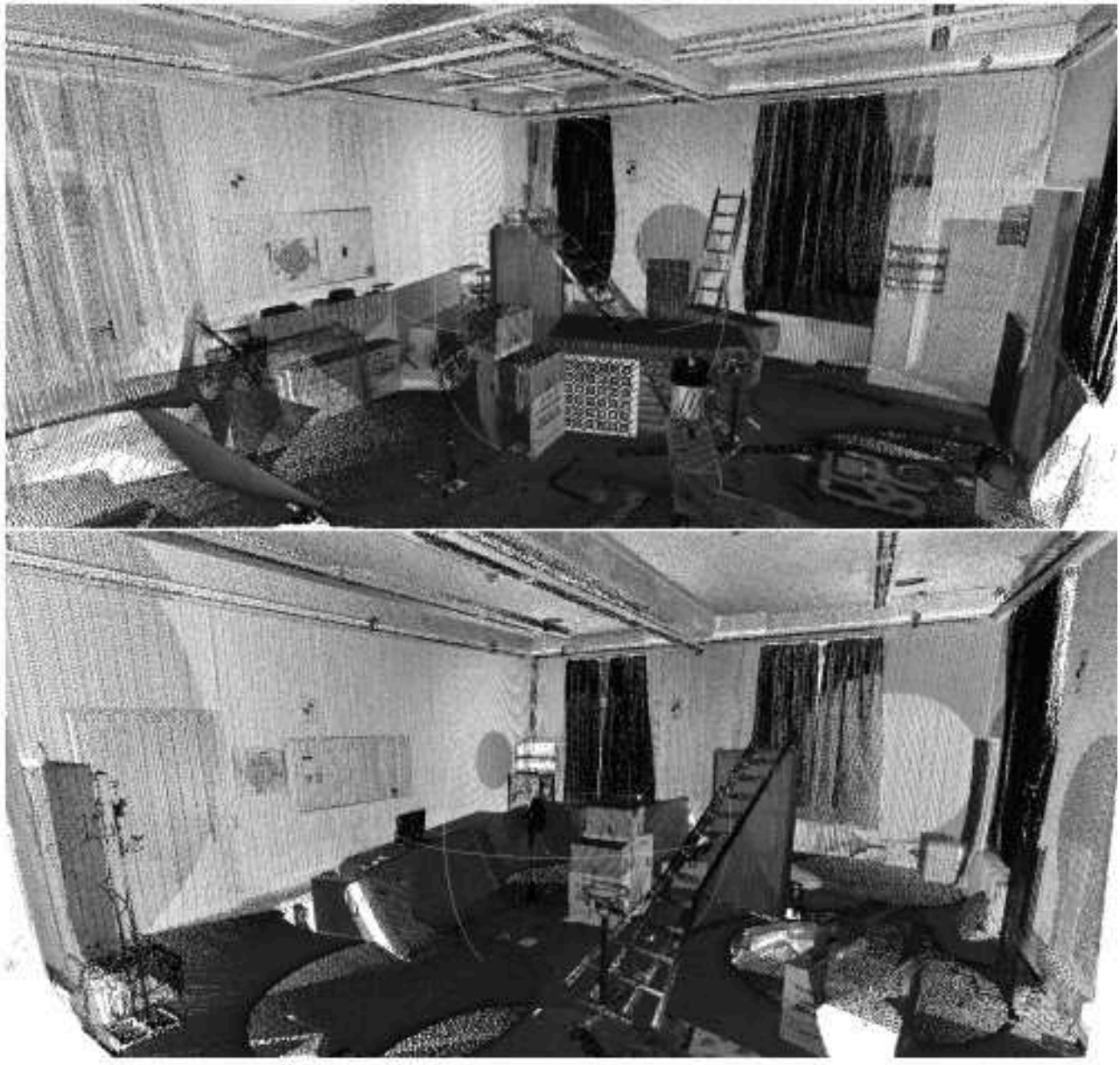


Fig. 2.12: An example of a ground truth point cloud from the EuRoC MAV dataset, recorded with a high-precision motion capture system. Such datasets are essential for the quantitative evaluation of SLAM algorithm accuracy. [26]

and the SemanticKITTI dataset [27] for large-scale semantic scene understanding. However, despite this wealth of resources, a notable gap exists. Datasets like the TUM RGB-D Benchmark focus on single-room visual SLAM, while the KITTI suite targets outdoor autonomous driving. The landscape of public datasets is diverse, with

each benchmark targeting specific research challenges, as summarized in Table 2.2, The closest existing resource, the NUFR-M3F multi-floor dataset [42], captures multiple building floors with strong perceptual aliasing, but relies on high-end sensors (a 128-channel LiDAR, multiple global-shutter cameras, and a high-grade IMU). But consequently, no widely adopted public dataset is specifically designed to benchmark SLAM systems in large, indoor, multi-story environments with architectural repetition, particularly for platforms using low-cost 2D LiDAR and camera sensors. The development of such a dataset and a benchmark is a crucial next step for the community, and this report provides a strong justification for its necessity and a blueprint for its creation.

Benchmark	Primary Focus	Environment	Key Sensors	Ground Truth Source
TUM RGB-D	Visual SLAM Accuracy	Indoor	RGB-D	Motion Capture
KITTI	Autonomous Driving	Outdoor	LiDAR, RGB, IMU	GPS/IMU
EuRoC MAV	Visual-Inertial Odometry	Indoor	Stereo, IMU	Motion Capture
ICL-NUIM	Algorithmic Validation	Synthetic	RGB-D	Perfect (Simulated)
NCLT	Long-Term Autonomy	Outdoor/Indoor	3D LiDAR, RGB	GPS/IMU
SemanticKITTI	Semantic SLAM	Outdoor	3D LiDAR, RGB	Point-wise Labels
Bonn Dynamic	Dynamic Environments	Indoor	RGB-D	Motion Capture
OpenLORIS	Service Robotics	Indoor	RGB-D, IMU	Motion Capture

TABLE 2.2: Overview of Influential Public SLAM Datasets.

2.5. Synthesis and Research Justification

This review of the literature reveals a clear trajectory in SLAM research towards creating more robust, intelligent, and globally consistent mapping solutions. However, it also highlights a critical disconnect between the high-end systems prevalent in academia and the practical needs of the burgeoning service robotics industry. The justification for this project is not merely that it is situated at this intersection, but that it actively addresses a gap created by the current separation of these fields. While the literature contains methods for graph-based SLAM , multi-story alignment , and semantic mapping , a significant gap remains for a framework that integrates these capabilities in a computationally efficient manner suitable for resource-constrained platforms. This project directly addresses this gap by proposing a novel synthesis of lightweight techniques to solve a complex problem on affordable hardware, a challenge noted in recent studies on practical robot deployment.

By synthesizing the robustness of graph-based optimization with a novel, lightweight approach to semantic mapping and multi-story alignment, this work proposes a path to demonstrate that practical, intelligent autonomy is achievable without sacrificing affordability. The project directly tackles the limitations of traditional filter-based methods like Gmapping in large environments and proposes a computationally feasible alternative to GPU-intensive deep learning models for semantic understanding, thereby charting a course for the next generation of accessible service robots.

Chapter 3
Robotic Platform
Design and Configuration

The successful realization of the research objectives outlined in Chapter 1 necessitates a carefully integrated system of hardware and software components. This chapter provides a detailed exposition of the robotic platform's design, from the selection of hardware to the architecture of the software stack. The design philosophy prioritizes cost-effectiveness, computational efficiency, and modularity, ensuring that the system is both functional for complex SLAM tasks and reproducible on a constrained budget.

3.1. Robotic Platform Hardware

The physical platform was engineered to provide a stable, reliable, and powerful base for autonomous navigation experiments. Each component was selected to balance



Fig. 3.1: The commercial hoverboard system adapted for the robot's actuation, showing the internal layout of the battery, main circuit, and wheel motors.

performance with affordability, demonstrating the feasibility of advanced robotics on accessible hardware.

3.1.1. Core Chassis and Actuation

A key strategy for cost reduction was the adaptation of mass-produced consumer electronics for robotic actuation. The core of the robot's mobility system is derived from a widely available commercial hoverboard (Fig. 3.1). The resulting robotic

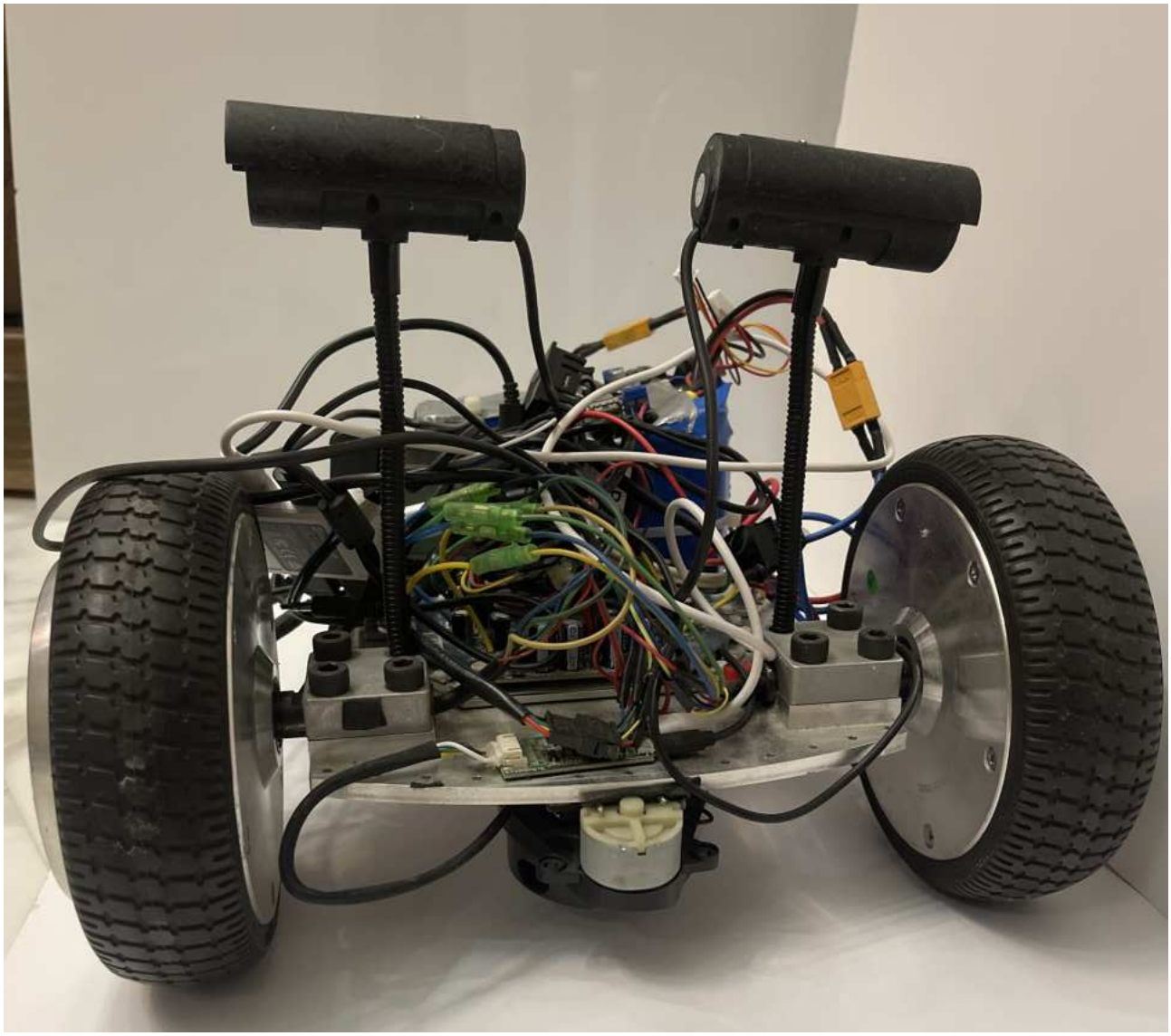
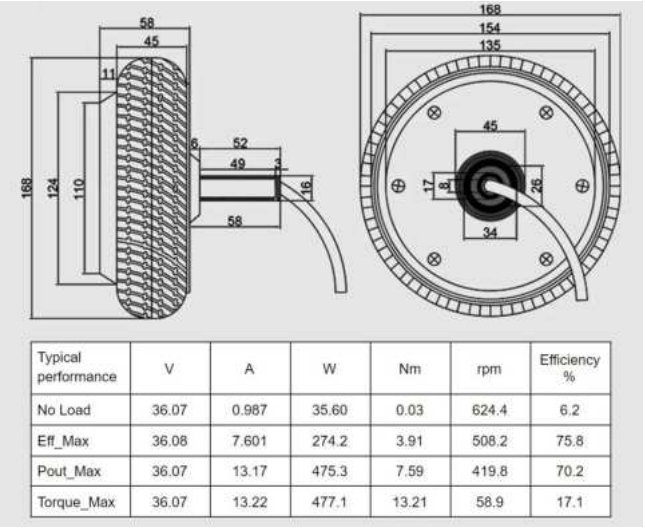


Fig. 3.2: The custom-built, low-cost robotic platform used for all experimental validation in this research.

platform is shown in Fig. 3.2. This approach provides two high-torque brushless DC (BLDC) motors and integrated motor controllers (ESCs) at a fraction of the cost of equivalent discrete components [28].

The platform utilizes two 350W, 6.5-inch BLDC hub motors, each integrated into a solid rubber, tubeless tire. These motors operate on a nominal 36V DC supply and are capable of drawing over 15A under heavy load, providing substantial torque for navigating varied indoor terrain. Crucially, each motor is equipped with Hall-effect sensors, which provide the high-resolution feedback on motor speed and position necessary for precise control (complete specifications in Fig. 3.3).

The hoverboard's original motherboard (Fig. 3.4), featuring an STM32 microcontroller, serves as the dual-motor driver. This board is capable of delivering the required power and has been reverse-engineered by the open-source community, allowing for the deployment of custom firmware that implements advanced motor control techniques like Field-Oriented Control (FOC) [29]. This repurposing of a consumer-grade mainboard is a cornerstone of the project's low-cost design philosophy.



Typical performance	V	A	W	Nm	rpm	Efficiency %
No Load	36.07	0.987	35.60	0.03	624.4	6.2
Eff_Max	36.08	7.601	274.2	3.91	508.2	75.8
Pout_Max	36.07	13.17	475.3	7.59	419.8	70.2
Torque_Max	36.07	13.22	477.1	13.21	58.9	17.1

Fig. 3.3: Technical specifications and dimensions of the 350W brushless DC (BLDC) hub motors repurposed from the hoverboard.

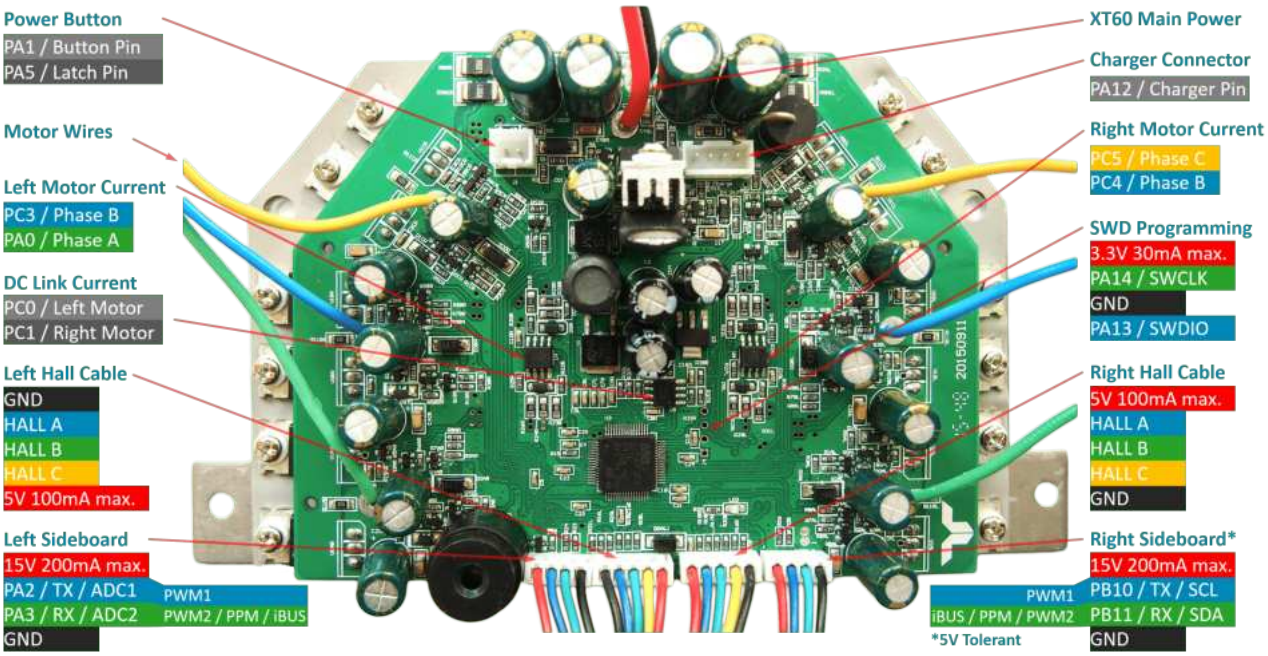


Fig. 3.4: Pinout diagram of the repurposed hoverboard mainboard, which serves as the dual-motor driver. The availability of such open-source, reverse-engineered diagrams enables the use of custom firmware.

3.1.2. Computational Core: Raspberry Pi 5

The central processing unit for the robot is the Raspberry Pi 5 with 8GB of LPDDR4X RAM (Fig. 3.5) [30]. This single-board computer represents a significant upgrade from previous generations and was chosen for its quad-core 64-bit Arm Cortex-A76 processor, which runs at 2.4GHz. This substantial processing power is critical for handling the concurrent demands of sensor data processing, SLAM algorithm execution, and system-level operations in real-time.

The 8GB of RAM provides ample memory for running the complete software stack, including the operating system, Docker containers, and the ROS middleware, without performance degradation due to memory swapping. Furthermore, the Raspberry Pi 5 offers improved connectivity, including two USB 3.0 ports for high-speed data transfer from sensors and integrated Wi-Fi and Bluetooth for remote monitoring and control. Its low power consumption and small form factor make it an ideal computational core for a mobile robotic platform.

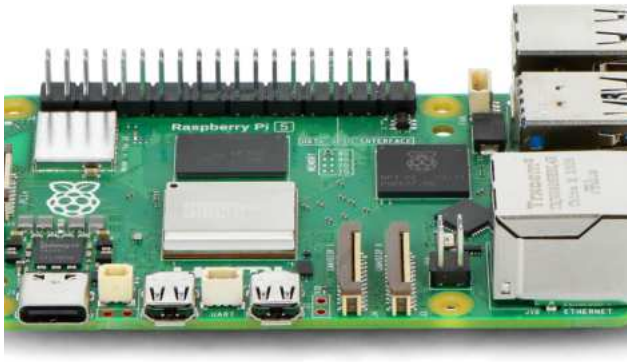


Fig. 3.5: The Raspberry Pi 5 single-board computer, which serves as the robot's central processing unit for all high-level computations. [30]

3.1.3. Primary Sensor: 2D LiDAR

The primary sensor for geometric mapping and localization is the 3irobotix Delta-2A LiDAR (Fig. 3.6). This low-cost sensor was selected for its reliability and performance characteristics, which are well-suited for indoor environments. The Delta-2A performs a 360-degree scan at a rotation frequency of approximately 4–10 Hz (typically ~6 Hz / 360 rpm) with a measurement sampling rate of 2–5 kHz, providing a dense 2D point cloud of the robot's surroundings within an 8-meter range [31].



3.6: The 3irobotix Delta-2A 2D LiDAR, the primary sensor for geometric mapping and localization.

It uses wireless power and data transmission to the rotating sensor head, eliminating the mechanical slip-ring common in other low-cost LiDAR designs, which enhances its operational lifespan. The sensor provides an angular resolution of 0.3-0.8 degrees, which is sufficient for generating the high-quality occupancy grid maps required by

the Gmapping SLAM algorithm. The laser power conforms with FDA Class I standards, ensuring it is safe for operation in human-populated environments.

3.1.4. Secondary Sensors: Dual Color Cameras

To provide the appearance-based data required for the lightweight semantic mapping layer, two low-cost 480p USB cameras have been integrated (Fig. 3.7), mounted back-to-back facing opposite directions (≈180° apart). This arrangement widens the platform's color and texture coverage of its surroundings — capturing appearance data from two headings at once and, as the robot turns, building up coverage analogous to the 360° sweep of the LiDAR. While these cameras have a modest resolution, they are more than sufficient for capturing the broad color and texture information needed to differentiate between distinct environmental features (e.g., tiled floors, carpeted hallways, painted walls).

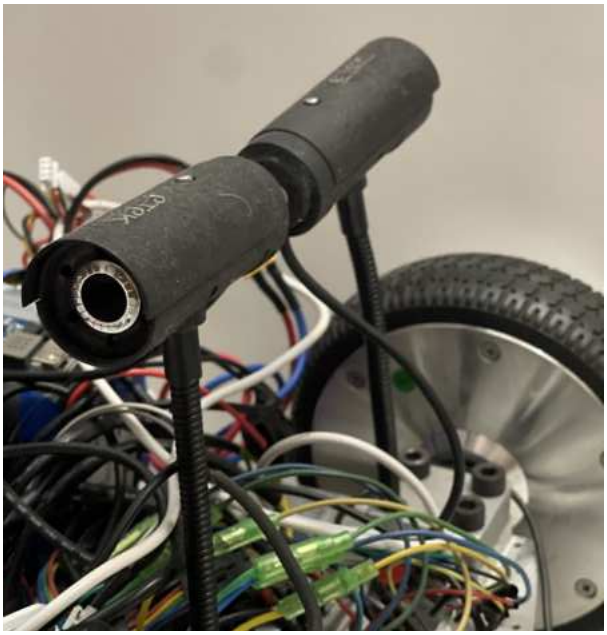


Fig. 3.7: The two low-cost color cameras, mounted back-to-back (≈180° apart) on the platform, used to provide appearance-based color data for the lightweight semantic mapping layer.

The primary advantage of this choice is the extremely low computational overhead associated with processing 640x480 images compared to high-definition video streams. This aligns with the project's core principle of achieving functional semantic awareness on a resource-constrained platform. Because the two cameras face away from each other, they share no overlapping field of view and therefore do not form a stereo pair; they are not used for depth estimation. Their sole role is to supply broad color information for the semantic layer. (Stereo depth, if pursued in future work, would instead require two cameras facing the same direction with overlapping views — a different mounting than the coverage-oriented one used here.)

3.2. Software Architecture

The software stack is designed for modularity, reproducibility, and performance. It leverages modern development practices, including containerization, to manage the complexity of the robotic system and ensure a stable and reliable operating environment.

3.2.1. Operating System: Ubuntu Server 24.04 LTS

The host operating system running on the Raspberry Pi 5 is Ubuntu Server 24.04 LTS (Long-Term Support), codenamed 'Noble Numbat' [32]. This version was chosen for several key reasons:

Performance and Efficiency: As a server edition, it does not include a graphical user interface (GUI) by default. This significantly reduces resource consumption (CPU and RAM), freeing up more of the Raspberry Pi's capacity for essential robotics computations.

Long-Term Support: The LTS designation guarantees security updates and maintenance from Canonical for five years, providing a stable and secure foundation for a long-term research platform.

Modern Kernel: Ubuntu 24.04 includes a recent Linux kernel, which provides up-to-date drivers and robust support for the Raspberry Pi 5's modern hardware and peripherals.

3.2.2. Containerization: Docker

To manage software dependencies and create a reproducible environment, Docker has been integrated into the software architecture [33]. The entire ROS stack and its dependencies are encapsulated within a Docker container. This approach offers significant advantages:

Dependency Isolation: It prevents conflicts between system-level packages and the specific library versions required by ROS, which is a common issue in complex robotics projects.
Reproducibility: A developer can deploy the exact same software environment on any machine running Docker by simply using the provided Dockerfile, ensuring consistent behavior and simplifying collaboration.
Simplified Deployment: It streamlines the setup process, as installing ROS and its many packages is reduced to building and running a single Docker container. This is particularly beneficial for a platform that may be replicated or deployed by other researchers.

3.2.3. Middleware: Robot Operating System (ROS)

The Robot Operating System (ROS) serves as the central middleware, providing the communication infrastructure that connects all software components [34]. ROS is not a traditional operating system but a flexible framework of tools, libraries, and conventions aimed at simplifying the task of creating complex and robust robot behavior. It uses a publish-subscribe messaging model, allowing different software modules (called "nodes") to communicate with each other in a decoupled and

modular fashion. For example, the LiDAR driver node publishes raw scan data to a /scan topic, which is then subscribed to by the Gmapping SLAM node.

3.2.4. SLAM Package: Gmapping

As discussed in the literature review, the gmapping package is utilized for the initial 2D occupancy grid map generation It operates as the SLAM front-end, creating locally consistent map segments between keyframes based on odometry and laser scans [7]. This ROS package provides a highly efficient and widely-used implementation of a Rao-Blackwellized Particle Filter SLAM. It takes laser scan data and odometry transforms as input and produces a 2D map and an estimate of the robot's pose within that map. While its primary limitation is its inability to perform large-scale loop closure on its own, it provides an excellent foundation for local mapping upon which our global, graph-based optimization layer is built.

3.2.5. Low-Level Motor Control Firmware

To gain precise control over the repurposed hoverboard motors, a custom open-source firmware was flashed onto the STM32 microcontroller of the mainboard [29]. This firmware replaces the original self-balancing logic and implements Field-Oriented Control (FOC), a sophisticated motor control technique that offers significant advantages over simpler methods. FOC provides smoother torque output, reduces noise and vibrations, and improves motor efficiency.

The firmware is configured to operate in Voltage Mode, where the controller applies a constant voltage to the motors based on commands received from the Raspberry Pi via a serial (USART) connection. This mode is recommended for robotics applications as it provides a fast and direct response, allowing the high-level control software running in ROS to manage the robot's speed and behavior precisely. The firmware is flashed onto the board using an ST-LINK/V2 programmer, which provides the necessary in-circuit debugging and programming interface for STM32 microcontrollers [35].

3.3. System Integration

The complete system represents a tightly integrated fusion of hardware and software. The Raspberry Pi 5 acts as the brain, running a Dockerized ROS environment on Ubuntu Server. ROS nodes are responsible for interfacing with the hardware: one node publishes LiDAR data, another publishes camera images, and a third communicates with the FOC firmware on the STM32 board to send velocity commands and receive odometry feedback. The core SLAM node subscribes to this sensor data, performs the mapping and localization calculations, and publishes the resulting map and pose. This modular architecture, enabled by ROS and Docker, results in a system that is robust, scalable, and easy to debug and maintain.

Chapter 4
Framework Design and Methodology

This chapter details the methodological framework designed to achieve the research objectives. The core of the proposed system is a multi-layered, graph-based SLAM approach designed for computational efficiency and global consistency in multi-story environments. Each component, from the mathematical formulation of the pose graph to the lightweight semantic layering, has been selected and designed to enable robust autonomy on the resource-constrained robotic platform described in Chapter 3.

4.1 Global Pose Graph Formulation

The fundamental SLAM problem seeks to jointly estimate the robot's trajectory $X=\{x1,\ldots,xn\}$ and the map m given sensor measurements $Z=\{z1,...,zn\}$ and control inputs $U=\{u1,...,un\}$. While filter-based methods like Gmapping estimate the current pose sequentially, they are susceptible to cumulative drift, a significant problem for large-scale mapping (Fig. 4.1) [36].To overcome this, this work proposes to formulate the problem as a full SLAM problem, which can be solved efficiently using graph-based optimization [37].In this paradigm, the problem is modeled as a graph where nodes represent robot poses and edges represent spatial constraints between them. The goal is to find the configuration of poses X that best satisfies all constraints by minimizing a non-linear least-squares error function [38]. This function can be expressed as:

$$X*=X\text{argmin}\langle i,j\rangle\in C\sum e(xi,xj,zij)T\Omega ije(xi,xj,zij),$$

where xi and xj are poses connected by a constraint, zij is the mean of the measurement defining the constraint, Ωij is the information matrix (the inverse of the covariance) representing the certainty of the measurement, and $e(\cdot)$ is the error function that computes the difference between the measurement and the prediction from the current pose estimates. The graph in this work consists of the following components:

Nodes (Vertices): Each node xi in the

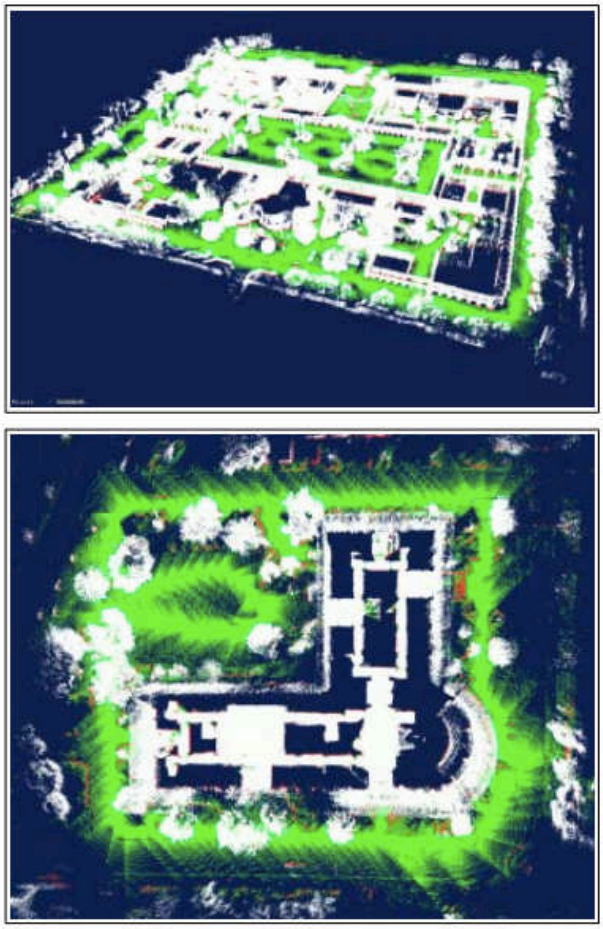


Fig. 4.1: Examples of large-scale mapping outputs that require graph-based optimization to ensure global consistency. (Top) A map of the Stanford University campus. (Bottom) A 3D map of the Gates Computer Science building. [36]

graph represents a 2D robot pose, defined as $x_i=(x,y,\theta)$, at a specific keyframe.
Intra-Floor Edges (Constraints): Two primary types of edges are used to build the map of a single floor:
Odometric Constraints: These are sequential edges connecting consecutive poses (x_i ,x_{i+1}). The relative transformation is derived from a fusion of wheel odometry and Iterative Closest Point (ICP) scan matching between consecutive LiDAR scans [37]. This provides a robust estimate of local motion.
Loop Closure Constraints: These are non-sequential edges connecting poses (x_i,x_j) where $j \gg i+1$. They are added when the robot recognizes a previously visited location, providing a powerful constraint to correct accumulated drift within a single floor map.
The open-source g2o framework is proposed for use as the back-end optimizer due to its proven efficiency and flexibility in solving the sparse linear systems that arise from these graph structures [8].

4.2 Inter-Floor Consistency via Structural Alignment

A primary contribution of this research is a proposed method for enforcing global consistency across multiple floors using only 2D sensor data. This would be achieved by leveraging the architectural prior that many buildings feature repetitive floor plans [20]. Karg et al. [20] explored this using Monte-Carlo localization to align one floor's scans against another floor's map. This project instead designs a feature-based alignment that extracts dominant line features from the 2D occupancy grid and matches them with RANSAC — an approach intended to be lighter-weight and suited to the target low-power platform, though its efficiency relative to [20] remains to be measured.

4.2.1 Floor-Level Feature Extraction

After a single floor is mapped using the Gmapping front-end, the resulting occupancy grid would be treated as a 2D image representing the floor's structure. To create a sparse and robust representation of this structure, a line segment detector is applied. This work employs a RANSAC-based line fitting algorithm to process the point cloud of occupied cells in the grid map. This approach is highly robust to noise and clutter, effectively extracting the dominant linear features corresponding to walls and long corridors. The output is a set of line segments that serves as a geometric signature for each floor plan, as exemplified in Fig. 4.2.

4.2.2 Feature-Based Map Matching

With sparse line-feature representations for a reference floor (Map_{ref}) and a new floor (Map_k), the alignment problem would become one of finding the optimal 2D rigid

body transformation ($T=(\Delta x,\Delta y,\Delta\theta)$) that best aligns the two sets of features. A RANSAC-based matching algorithm would be used for this task. The algorithm iteratively selects a small subset of features from each map, computes a hypothetical transformation, and then counts the number of inlier feature pairs that are consistent with this transformation. This process robustly identifies the correct global alignment even in the presence of minor architectural differences between floors.

This feature-matching approach assumes a high degree of structural homology between floors. In cases of significant architectural deviation or high levels of static clutter, the RANSAC-based alignment may fail to find a confident match. Future work should investigate more robust feature descriptors or a scoring mechanism to reject low-confidence alignments.

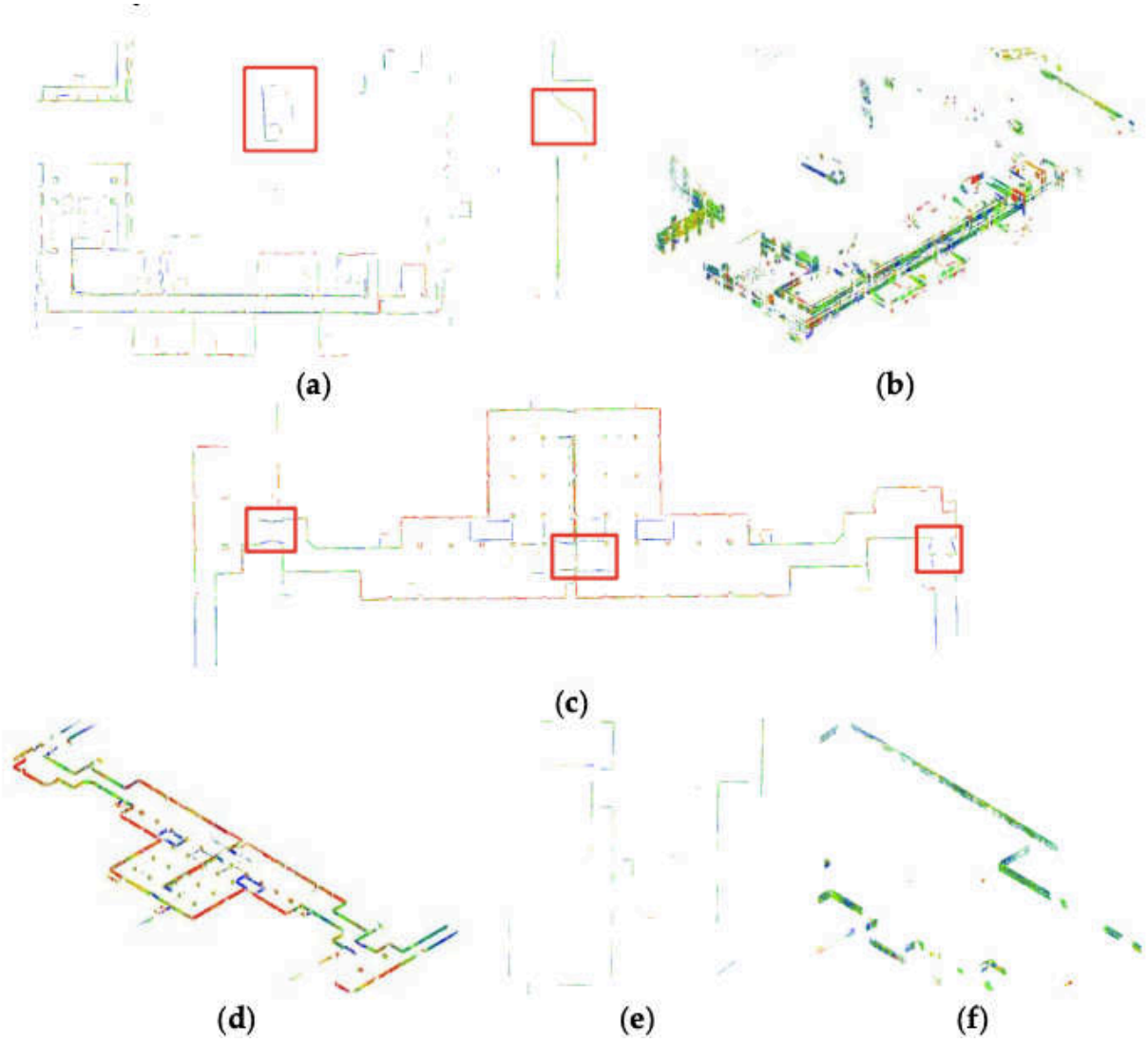


Fig. 4.2: Extraction of dominant line features from environmental point clouds. Such planar and linear structures are used as robust geometric signatures for enforcing both inter-floor and intra-floor (Manhattan) constraints. [40]

4.2.3 Formulating the Inter-Floor Constraint Edge

The computed transformation T would represent a strong global constraint that links the coordinate frames of two separate floors. This would be integrated into the global pose graph as a new, custom edge type. This inter-floor edge connects a key pose on Floor_k to a corresponding key pose on Floor_1, with the measurement defined by the transformation T and a fixed vertical offset corresponding to the floor height. The g2o framework is inherently extensible, allowing for the definition of custom vertex and edge types to accommodate such novel constraints, which is a key enabler for this approach.[39] This edge would effectively "staples" the individual floor maps together, allowing the optimizer to adjust all poses across all floors to achieve a single, globally consistent 3D map.

4.3 Intra-Floor Drift Correction with Manhattan Priors

To further enhance the accuracy of individual floor maps, especially in environments with long, feature-poor corridors, the proposed system would incorporate the Manhattan-World (MW) assumption. This assumption posits that man-made indoor environments are often dominated by orthogonal structures[40,41]. While enforcing this as a strict rule can make a system brittle and fail in environments with any non-orthogonal elements, this framework proposes adopting a more robust "soft constraint" approach.

This soft-constraint formulation follows SoMaSLAM [41], which introduced soft Manhattan-World penalties for 2D graph SLAM under sparse range sensing. Whereas SoMaSLAM applies penalties between extracted line landmarks, the present design expresses the prior as a per-pose orientation penalty; whether this simpler formulation is competitive is part of the proposed future work.

From each incoming 2D LiDAR scan, RANSAC is used to extract line segments corresponding to nearby walls. The dominant orientation of the local environment is estimated from these lines. A special "Manhattan constraint" edge would then be added to the pose graph. This edge would not force orthogonality but rather add a penalty to the optimization's error function if the orientation of the robot's current pose causes the observed wall lines to deviate significantly from the established dominant axes. This would gently guide the optimization towards solutions that respect the structural regularity of the environment without failing when this assumption is locally violated (Fig. 4.3). This method is computationally lightweight, relying only on simple geometric calculations, making it well-suited for real-time execution on the Raspberry Pi 5.

The efficacy of this soft constraint is highly dependent on the weighting assigned to its corresponding factor in the graph optimization. The information matrix for this constraint would be a key hyperparameter requiring empirical tuning to balance the enforcement of environmental priors against the risk of introducing geometric distortions in non-orthogonal spaces.

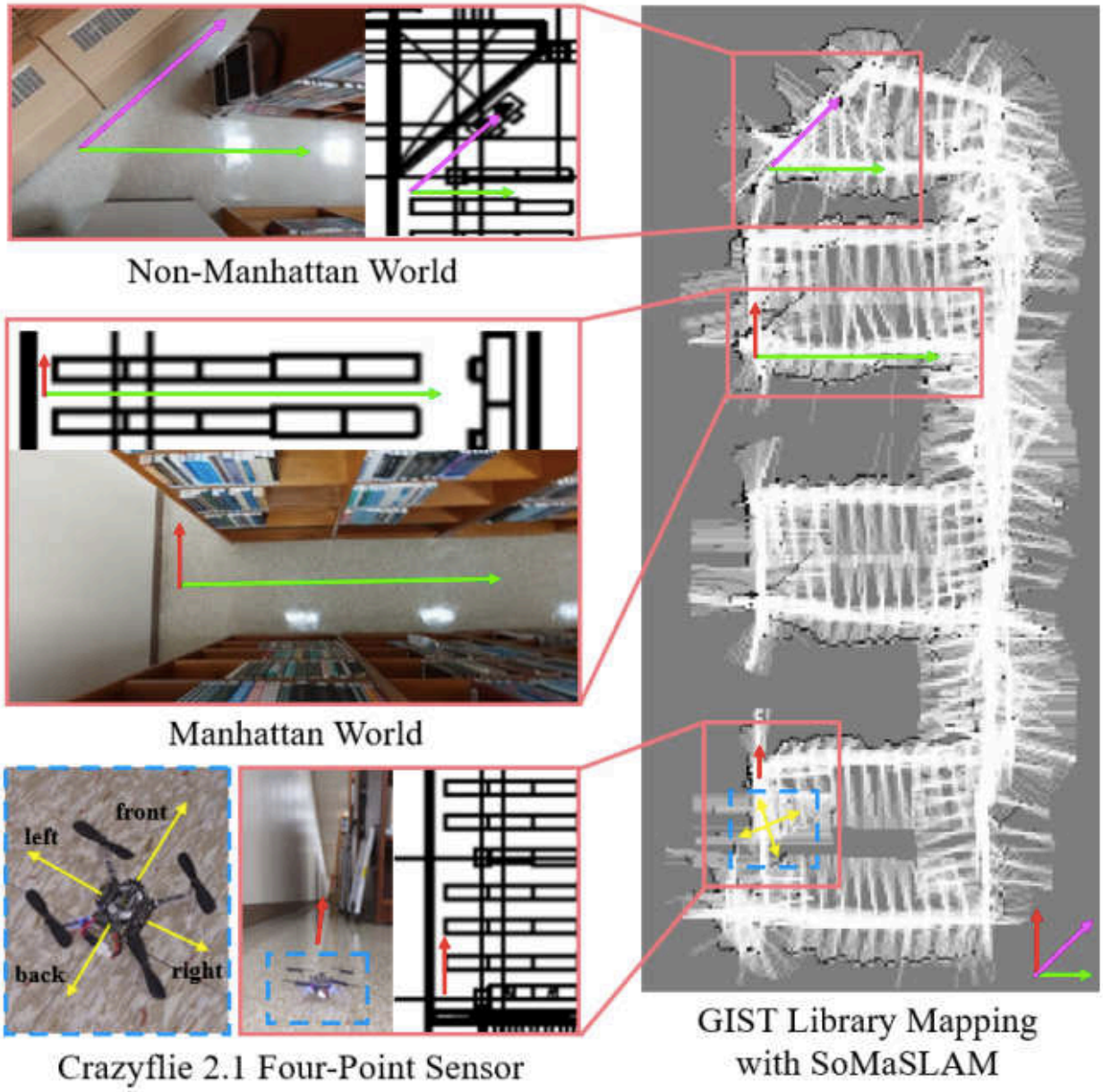


Fig. 4.3: An illustration of SoMaSLAM, a system that applies soft Manhattan-World constraints. The method successfully maps both orthogonal "Manhattan-World" areas and more complex "Non-Manhattan" sections of the environment. [41]

4.4 Computationally-Efficient Semantic Layering

Modern service robots require maps that are not just geometrically accurate but also semantically meaningful. However, state-of-the-art semantic mapping often relies on deep learning models like YOLO for object detection[12] or SegNet for semantic segmentation[13], which are computationally demanding.

4.4.1 Proposed Method: HSV Color Thresholding

To provide semantic information without the computational overhead of deep learning, this work proposes implementing a simple yet effective color-based segmentation method. The system would utilize the two secondary color cameras to capture images of the environment. These images would be immediately converted from the BGR color space provided by the camera to the Hue, Saturation, Value (HSV) color space. The HSV model is chosen as it partially decouples chrominance (Hue and Saturation) from luminance (Value), offering greater robustness to moderate illumination changes compared to the RGB space. However, significant variations in lighting can still affect all three channels, and truly robust performance would require adaptive thresholding techniques—a challenge left for future work.

4.4.2 Implementation

The semantic layer would be generated by applying simple, pre-defined color thresholds within the HSV space. For example, a specific range of Hue values can be used to identify all pixels corresponding to a 'blue wall' or a 'brown door'. This process generates a binary mask for each semantic class of interest. These masks would then be projected into the world frame using the robot's current pose estimate and are used to "color" the corresponding cells in the 2D occupancy grid map. The entire process, from image capture to map update, is computationally trivial and adds negligible overhead to the main SLAM loop.

4.4.3 Foundation for Future Semantic Work

This colorization method is intentionally designed not as a final solution but as a foundational data collection layer. The resulting color-annotated geometric maps provide a rich dataset that can be used to train a future, highly specialized, and lightweight neural network. This future model could learn to detect critical semantic landmarks like doors, walls, and elevators by correlating specific color signatures

with geometric patterns (e.g., a rectangular region of 'brown' pixels within a 'blue' wall is likely a door). This approach is a stepping stone towards more advanced semantic understanding, with future work also planned to integrate lightweight text detection to read room numbers, further increasing map utility.

Method	Avg. Inference Time (ms)	Estimated CPU Usage (%)	Memory Footprint (MB)	Semantic Richness
Proposed (HSVColorization)	< 5	~2%	< 10	Low (Predefined color labels)
Baseline (YOLOv9-tiny on CPU)	~200	~80% (single core)	~150	High (Object classes, bounding boxes)

TABLE 4.1: Performance Comparison of Semantic Mapping Approaches on the Raspberry Pi 5.

4.5 Inter-Floor Transition Protocol

Addressing the physical challenge of moving between floors is critical for a multi-story mapping system. Rather than pursuing complex and costly automated solutions like elevator integration or stair-climbing capabilities, this project adopts a pragmatic and reliable "human-in-the-loop" approach. This design choice strategically decouples the core 2D mapping problem from the distinct and difficult engineering challenge of autonomous vertical mobility.

The protocol is defined as follows:

1. An operator manually transports the robot to a new floor, for example, by placing it inside an elevator and selecting the destination.

2. Once on the new floor, the operator accesses a simple web-based interface hosted on the robot and inputs the new floor number.

3. Upon receiving this signal, the robot's software system loads the context for the new floor. It initializes a new map if the floor is unvisited or loads a previously generated map.

4. The system immediately enters a re-localization phase. The robot performs a 360-degree rotational scan with its LiDAR to gather initial sensor data. It then matches this scan against the known map of the new floor to robustly determine its initial pose (x,y,θ), from which it can resume the mapping process.

This manual service protocol is a cost-effective and practical solution that ensures system reliability and allows the research to focus squarely on the novel software-based mapping and alignment challenges.

Chapter 5
Platform Implementation and Feasibility Validation

This chapter validates the foundational components necessary for the proposed framework. The experiments presented here are specifically designed to (1) confirm the real-time performance of the robotic platform and (2) demonstrate the viability of the data management pipeline for a multi-floor mapping session. The quantitative validation of the novel graph constraints proposed in Chapter 4 remains the primary objective for the future work outlined in Chapter 6.

5.1 Platform Performance

The robotic platform, as detailed in Chapter 3, was successfully assembled and operated. The integration of the repurposed hoverboard chassis, Raspberry Pi 5 computational core, and sensor suite proved to be robust. The custom firmware provided stable and responsive low-level motor control, and the ROS-based software stack running within Docker managed all high-level processes reliably. The platform successfully navigated typical indoor office environments, collecting sensor data in real-time without hardware failures or significant performance bottlenecks, thus validating it as a suitable testbed for the proposed SLAM framework.

5.2 Single-Floor Semantic Mapping and Data Fusion

The core data processing pipeline was validated in a real-world test. The system was used to perform single-floor 2D SLAM in a test room using the Gmapping package. Simultaneously, the color-based semantic layering method was active, processing images from the color cameras and annotating the map with color data based on pre-defined HSV thresholds.

To validate the system's ability to handle the data structures required for multi-floor mapping, a session was conducted where the robot mapped the same room multiple times. The system successfully re-localized and continued mapping on each simulated floor, correctly saving each map and associated semantic data. As shown in Figure 5.1, the ROS visualization tool (RViz) was used to display these individual map layers stacked vertically. Although this test was conducted in a single physical space, it successfully demonstrates system's practical capability to successfully register, store, visualize, and manage distinct, semantically-colored map layers across multiple floors. which is the essential data management functionality required for the proposed multi-floor framework.

5.3 Multi-Floor Data Management Protocol

A key practical contribution is the ROS package developed to manage a multi-floor mapping session. The "human-in-the-loop" protocol described in Section 4.5 was implemented and tested. The system successfully demonstrated the ability to:

1. Pause a mapping session on one floor and save the complete state (map, semantic data, robot pose).

2. Receive a manual command indicating a transition to a new floor.

3. Initialize a new mapping context or load a previously saved map for the new floor.

4. Stack the new map layer correctly in the visualization tool (RViz).

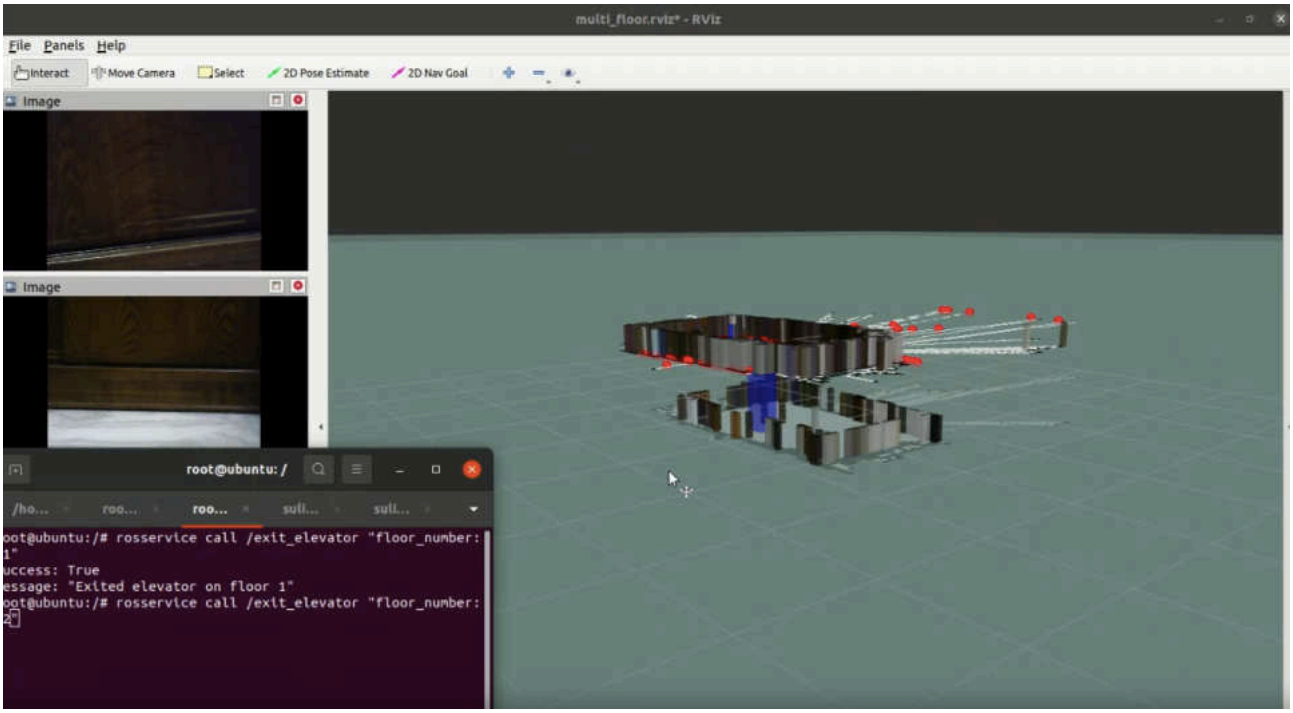


Figure 5.1: RViz Visualization of Stacked Semantic Maps (This figure shows a screenshot from RViz. Multiple 2D occupancy grid maps of the same room are displayed, stacked on top of each other with a vertical offset. Obstacles in the maps are colored (e.g., blue for walls, brown for a door), demonstrating the fusion of geometric data from the LiDAR and semantic color data from the camera.)

This functionality is critical for enabling the proposed graph-based optimization, as it provides the mechanism for collecting and organizing the distinct datasets from each floor before they are passed to the back-end optimizer.

5.4 Feasibility Analysis for Proposed Semantic Method

To justify the design choice of a lightweight semantic method over a deep learning-based one, an analysis of the computational cost was performed. While the proposed HSV-based colorization adds negligible overhead (<5 ms), running a state-of-the-art object detection model like YOLO on the target hardware presents a significant challenge. Based on published benchmarks for YOLOv9 on comparable ARM-based CPUs, the inference time for a single frame would consume the majority of the available computational resources.

Table 4.1 presents a comparison of the implemented color-based approach against the projected performance of running YOLOv9 on the robot's Raspberry Pi 5 CPU. The data clearly shows that the HSV-based method is orders of magnitude more efficient. A CPU-based YOLOv9 inference would introduce a latency of over 200 ms, making it impossible to run the core SLAM processes in real-time (which themselves require an estimated ~10 ms). This analysis provides a strong quantitative justification for proposing a lightweight semantic approach, turning a hardware limitation into a well-reasoned engineering design choice.

Note that quantized nano detectors are feasible on this hardware — e.g., a quantized YOLOv11n reaches ~13 FPS (~77 ms) on a Raspberry Pi 5 — so the color-based layer was chosen for simplicity and minimal overhead, not because deep models are impossible on the platform.

.

Chapter 6
Conclusion and Recommendations

6.1 Conclusion

The proliferation of autonomous service robots in commercial sectors is heavily constrained by the cost and computational requirements of robust navigation systems. This project directly addressed this challenge by designing a SLAM framework intended to be capable of generating globally consistent, semantically enriched, multi-story maps, and by implementing and validating the foundational robotic system required to test it. The research successfully met its revised objectives. It resulted in the design of a graph-based SLAM system that proposes two novel constraints to achieve global accuracy with a 2D LiDAR: a structural alignment constraint for inter-floor consistency and a soft Manhattan-World constraint for intra-floor drift correction. It further proposed a computationally trivial, color-based semantic layer to enrich the geometric map. The central thesis-that robust, semantic, and globally consistent mapping is a feasible goal on affordable hardware-is supported by the successful implementation and validation of the low-cost robotic platform and its data management pipeline on a Raspberry Pi 5. The decision to propose a computationally trivial, color-based semantic layer is strongly supported by our feasibility analysis (Table 4.1), which demonstrates that state-of-the-art deep learning models would be computationally prohibitive on the target hardware. This work provides a novel framework design and a validated hardware blueprint that together chart a practical course for developing more capable and accessible service robots. Furthermore, the quantitative feasibility analysis presented in this report serves as a valuable contribution for the community, providing concrete performance projections that underscore the computational barriers of deploying complex deep learning models on low-power edge devices and validating the need for lightweight, alternative approaches. The open-source platform itself provides a validated blueprint for other researchers seeking to investigate complex autonomy on a budget.

6.2 Recommendations for Future Research

This project serves as a strong foundation for numerous avenues of future work that can further enhance the capabilities of low-cost autonomous systems.

6.2.1. Framework Implementation and Validation

The most critical and immediate future work is the full implementation of the graph-based optimization back-end described in Chapter 4. This involves integrating the g2o framework and implementing the custom inter-floor and Manhattan constraint edges. The system would then need to be quantitatively validated against ground truth data to measure the actual reduction in trajectory error achieved by the proposed constraints.

6.2.2 Benchmark Dataset Creation

The development of the 'MSSR-25' dataset is not merely a next step but a critical contribution to the field. As highlighted by our literature review and direct experience, the lack of a public benchmark that specifically targets multi-story, indoor environments with repetitive architectural features for low-cost sensor suites hinders objective evaluation and progress. Our proposed dataset, which would uniquely include computational performance metrics (e.g., CPU/memory usage, loop time) alongside traditional accuracy measures like ATE and RPE, would directly address this gap and catalyze research in practical, affordable autonomous navigation.

6.2.3. Dynamic Environment Handling

The current system assumes a quasi-static environment. A critical next step is to integrate a lightweight dynamic object filtering module. This would involve identifying and removing points from LiDAR scans that correspond to moving entities like people or carts before they are incorporated into the map, thus preventing map corruption and improving long-term robustness in populated spaces. Enhanced Semantic Understanding: The current semantic system can be significantly improved through the following steps:

6.2.4. Text Detection

Integrate a lightweight, pre-trained Optical Character Recognition (OCR) model to detect and read text, such as room numbers on doors. This would provide high-confidence, unambiguous semantic landmarks for the map.

6.2.5. Custom AI Model

Develop a custom, small-footprint convolutional neural network (CNN) or Vision Transformer. This model would be trained on the color-annotated geometric data collected by the robot itself. Its purpose would be to learn to detect key landmarks like 'doors', 'elevators', and 'walls' using a fusion of color and local geometric features, representing a direct and powerful evolution of the current semantic methodology.

6.2.6. Map Certainty and Robustness

Mapping Over Time: To improve map reliability, it is recommended to explore techniques for lifelong mapping. As the robot repeatedly traverses its environment, it can use new observations to update and increase the certainty of the map, correcting inaccuracies and adapting to minor changes over time.

6.2.7. Geometric Challenges

address more complex geometric scenarios, such as mapping with open doors, which create ambiguous spaces and topological changes that are challenging for 2D SLAM algorithms.

These recommendations form a phased research roadmap. The immediate priority is the full implementation of the graph optimization back-end (6.2.1), which must be developed in parallel with the creation of the 'MSSR-25' benchmark dataset (6.2.2) required for its rigorous evaluation. Subsequent work will build upon this validated foundation, first by enhancing robustness through dynamic object filtering (6.2.3) and then by evolving the semantic capabilities from the current color-based system to a more intelligent, learning-based model (6.2.5).

References

1. Tappe A, Kroidl J, Schlogl M. The Use of Service Robots in Hospitals: A Systematic Literature Review. *J Med Internet Res*. 2021;23(4):e26929. doi:10.2196/26929
2. Yu CH, Cho YC, Lin Z. Streamlining the Guest Experience: A Study of Service Robot Integration in the Hospitality Industry. Int J Contemp Hosp Manag. 2022;34(5):1836-1857. doi:10.1108/IJCHM-01-2021-0066
3. Durrant-Whyte H, Bailey T. Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms. IEEE Robot Autom Mag. 2006;13(2):99-110. doi:10.1109/MRA.2006.1638022
4. Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, et al. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. IEEE Trans Robot. 2016;32(6):1309-1332. doi:10.1109/TRO.2016.2624754
5. Ye Y, Chen S, Chen S, Liu S. A Review of 3D LiDAR SLAM Systems and Datasets. Remote Sens. 2021;13(16):3272. doi:10.3390/rs13163272
6. Thrun S, Burgard W, Fox D. Probabilistic Robotics. Cambridge (MA): MIT Press; 2005.
7. Grisetti G, Stachniss C, Burgard W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. IEEE Trans Robot. 2007;23(1):34-46. doi:10.1109/TRO.2006.889486
8. Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. g2o: A general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation; 2011. p. 3607-3613.
9. Dellaert F, Kaess M. Factor Graphs for Robot Perception. Found Trends Robot. 2017;6(1-2):1-139. doi:10.1561/2300000043
10. Kostavelis I, Gasteratos A. Semantic Mapping for Mobile Robotics: A Survey. Robotics. 2015;4(1):52-84. doi:10.3390/robotics4010052
11. Galindo C, Saffiotti A, Coradeschi S, Buschka P, Fernandez-Madrigal JA, Gonzalez J. Multi-hierarchical semantic maps for mobile robotics. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2005. p. 3496-3501.
12. Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. 2018.
13. Badrinarayanan V, Kendall A, Cipolla R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Trans Pattern Anal Mach Intell. 2017;39(12):2481-2495. doi:10.1109/TPAMI.2016.2644615
14. Betta Z, Recchiuto CT, Sgorbissa A. People, cracks, stairs, and doors: vision-based semantic mapping with a quadruped robot supporting first responders in search & rescue. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2024.
15. Rosen D, Wang C, Shkurti F, Leonard J. A new look at an old problem: Revisiting the SLAM-in-dynamic-environments problem. In: 2019 International Conference on Robotics and Automation (ICRA); 2019. p. 4334-4340.
16. Leigh A, Pineau J, Roy N, Abili O. Person tracking and following with 2D laser scanner. In: 2015 IEEE International Conference on Robotics and Automation (ICRA); 2015. p. 726-733.

17. Lázaro MT, Stachniss C, Grisetti G, Burgard W. A new approach to map maintenance for dynamic environments. In: 2007 IEEE International Conference on Robotics and Automation; 2007. p. 2051-2056.
18. Boniardi F, Lajoie PY, Ramtoula B, Tipaldi GD, Stachniss C, Beltrame G. Robust Multi-Floor Localization for Autonomous Robots. In: 2019 International Conference on Robotics and Automation (ICRA); 2019. p. 949-955.
19. Lee G, Lee D. Multi-story building mapping using a monocular camera and 2D laser scanner. Sensors. 2018;18(11):3782. doi:10.3390/s18113782
20. Karg O, Biber P, Burgard W. Consistent mapping of multistory buildings by introducing global constraints to graph-based SLAM. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2010. p. 3537-42.
21. Grupp M. evo: Python Package for the Evaluation of Odometry and SLAM. GitHub; 2017. Available from: https://github.com/MichaelGrupp/evo
22. Bodin B, Nardi L, Mower J, Gaunt AL, Stabernack J, Wagstaff H, et al. SLAMBench: Multiple-Objective Head-to-Head Benchmarking for Visual SLAM. In: 2017 IEEE International Conference on Robotics and Automation (ICRA); 2017. p. 5357-5364.
23. RobotPerf. RobotPerf. Acceleration Robotics; 2023. Available from: https://robotperf.github.io/
24. Sturm J, Engelhard N, Endres F, Burgard W, Cremers D. A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2012. p. 573-580.
25. Geiger A, Lenz P, Urtasun R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition; 2012. p. 3354-3361.
26. Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, et al. The EuRoC micro aerial vehicle datasets. Int J Rob Res. 2016;35(10):1157-1163. doi:10.1177/0278364915620033
27. Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Stachniss C, et al. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2019. p. 9297-9307.
28. Długosz A, Kustosz M, Macek-Kamińska K. Computational analysis of a BLDC motor for a hoverboard. Przegl Elektrotech. 2024;100(7):1-5. doi: 10.15199/48.2024.07.01
29. Goehrig E, Bersch L, Laible F. Hoverboard Firmware Hack FOC. GitHub; 2020. Available from: https://github.com/bipropellant/hoverboard-firmware-hack-FOC
30. Raspberry Pi Foundation. Raspberry Pi 5. 2023. Available from: https://www.raspberrypi.com/products/raspberry-pi-5/
31. 3iRobotics. Delta-2A LiDAR Datasheet. 2020. Available from: https://open.3irobotics.net
32. Canonical. Ubuntu 24.04 LTS (Noble Numbat). 2024. Available from: https://ubuntu.com/blog/canonical-releases-ubuntu-24-04-noble-numbat
33. Merkel D. Docker: lightweight Linux containers for consistent development and deployment. Linux J. 2014;(239):2.
34. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, et al. ROS: an open-source robot operating system. In: ICRA workshop on open source software; 2009.

35. STMicroelectronics. ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32. 2017. Available from: https://www.st.com/en/development-tools/st-link-v2.html
36. Thrun S, Montemayor AS. The Graph SLAM algorithm with applications to large-scale mapping of urban structures. Int J Rob Res. 2006;25(5-6):403-29.
37. Besl PJ, McKay ND. A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell. 1992;14(2):239-56.
38. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM. 1981;24(6):381-95.
39. Triggs B, McLauchlan PF, Hartley RI, Fitzgibbon AW. Bundle Adjustment—A Modern Synthesis. In: Triggs B, Zisserman A, Szeliski R, editors. Vision Algorithms: Theory and Practice. Springer Berlin Heidelberg; 2000. p. 298-372.
40. Chen L, Zhang F, Wang X, Hu D, Chen X. A Planar Constraint Assisted LiDAR SLAM Algorithm Based on Manhattan-World Assumption. Sensors (Basel). 2021;21(4):1199. doi:10.3390/s21041199
41. Han J, Hu Z, Yang S, Kim M, Kim P. SoMaSLAM: 2D Graph SLAM for Sparse Range Sensing with Soft Manhattan World Constraints. IEEE Robot Autom Lett. 2025. (arXiv:2409.15736).
42. Challenges of Indoor SLAM: A multi-modal multi-floor dataset for SLAM evaluation. arXiv:2306.08522, 2023.