ROS-Based 2D SLAM and Indoor Mapping Using Rawseeds Laser Data

Mohammad Althaf Syed
CWID: 20034265
CPE 521-A Autonomous Mobile Robotic Systems
Stevens Institute of Technology

***Abstract*—This project focuses on the implementation of a two-dimensional Simultaneous Localization and Mapping (SLAM) system using a single front-mounted laser scanner within the Robot Operating System (ROS) framework. Experimental sensor data provided in CSV format, consisting of odometry and front laser range measurements, were first converted into a ROS bag file to enable offline SLAM processing. The gmapping algorithm was applied to estimate the robot's trajectory while incrementally constructing an occupancy grid map of the environment. Mapping performance was evaluated under default gmapping settings as well as a refined configuration designed to improve scan matching and reduce mapping noise. Visualization and validation of the SLAM process were performed using RViz, and the final occupancy grid was exported in standard PGM and YAML formats for documentation and analysis. The resulting maps demonstrate that front-laser-based gmapping can reliably produce a coherent and structurally consistent representation of the environment, with noticeable improvements observed after parameter refinement. This project demonstrates a complete and reproducible SLAM pipeline and highlights the effectiveness of gmapping for 2D mapping using a single forward-facing laser sensor.**

***Index Terms*—SLAM, GMapping, ROS Noetic, Occupancy Grid Mapping, Laser SLAM**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental capability for autonomous mobile robots operating in unknown environments, enabling simultaneous estimation of robot pose and construction of an environmental map using onboard sensors. Accurate SLAM is essential for tasks such as navigation, exploration, and environment understanding, and remains an active area of robotics research.

In this project, a complete two-dimensional SLAM pipeline was implemented using the Robot Operating System (ROS) Noetic. The objective was to generate an occupancy grid map of an indoor environment using a single front-mounted laser scanner and wheel odometry data. The SLAM system was evaluated using real-world benchmark data from the Rawseeds Project (Laser SLAM – Bicocca 2009-02-25b), enabling controlled and repeatable offline mapping experiments.

The `gmapping` algorithm was selected due to its effectiveness for laser-based 2D SLAM using a Rao–Blackwellized particle filter framework. Both default and refined parameter configurations were evaluated to study their impact on map quality. The resulting occupancy grid maps demonstrate the importance of parameter tuning in improving scan alignment and reducing mapping artifacts.

II. System Setup: Ubuntu and ROS Noetic Installation Using UTM on Apple Silicon (Mac M4)

A. *Platform and Virtualization Environment*

The project was developed on an Apple Silicon-based Mac M4 system. Since ROS Noetic is designed to run natively on Linux, a virtualized Ubuntu environment was created using *UTM*, a lightweight virtualization tool optimized for Apple Silicon architectures. UTM was selected to provide a stable and isolated Linux environment while maintaining compatibility with ROS Noetic and its dependencies.

An Ubuntu Linux distribution compatible with the ARM architecture was installed within the UTM virtual machine. This virtualized setup enabled full access to the Linux terminal, package management tools, and system libraries required for ROS development, while allowing the project to be executed entirely on macOS hardware.

B. *ROS Noetic Installation and Configuration*

ROS Noetic was installed inside the Ubuntu virtual machine following the official ROS installation guidelines. The ROS Noetic distribution was chosen in accordance with the project requirements specified in the final assignment instructions. After installation, the ROS environment was configured by sourcing the appropriate setup files to ensure that ROS packages, nodes, and tools were accessible from the terminal.

Additional ROS packages required for this project, including `gmapping`, `map_server`, and visualization tools such as `rviz`, were installed using the Ubuntu package manager. The ROS master (`roscore`) was used to manage communication between nodes during SLAM execution.

C. *Development Environment Validation*

The correctness of the software setup was verified by launching core ROS services, visualizing topics in RViz, and successfully replaying ROS bag files using simulated time. The UTM-based Ubuntu environment demonstrated stable performance during offline bag playback and SLAM execution, confirming its suitability for the implementation and evaluation of the SLAM pipeline.

III. SLAM Algorithm Overview and Methodology

A. Overview of Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) enables a mobile robot to estimate its pose while simultaneously constructing a map of an unknown environment using noisy sensor measurements. These two tasks are inherently coupled, as accurate localization depends on a reliable map, while map construction requires knowledge of the robot's position. In indoor environments, SLAM is commonly performed using two-dimensional laser range data and wheel odometry, making it well suited for ground robots operating on planar surfaces.

In this project, an offline two-dimensional laser-based SLAM approach is adopted to generate an occupancy grid map of the environment. Offline processing of recorded sensor data allows controlled, repeatable experiments and consistent evaluation of mapping performance.

B. GMapping Algorithm (Rao–Blackwellized Particle Filter)

The SLAM implementation is based on the `gmapping` algorithm available in ROS. Gmapping employs a Rao– Blackwellized Particle Filter (RBPF) to estimate the robot trajectory while incrementally constructing an occupancy grid map. The RBPF framework factorizes the full SLAM posterior distribution into a robot trajectory estimation term and a conditional map estimation term, expressed as

$$P(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = P(m \mid x_{1:t}, z_{1:t}) \cdot P(x_{1:t} \mid z_{1:t}, u_{1:t}), \tag{1}$$

where $x_{1:t}$ represents the robot pose history, m denotes the occupancy grid map, $z_{1:t}$ corresponds to laser scan measurements, and $u_{1:t}$ represents odometry inputs.

The robot trajectory posterior is estimated using a particle filter, where each particle represents a possible pose hypothesis. Given a fixed trajectory, the corresponding map can be estimated independently using standard occupancy grid mapping, significantly reducing computational complexity.

C. Core Algorithmic Mechanism

For each incoming sensor measurement, the gmapping algorithm follows a recursive three-step process:

- **Prediction**: Odometry measurements propagate each particle forward using the robot motion model.
- **Weighting (Scan Matching)**: The latest laser scan is aligned with the particle's map, and a likelihood score is assigned based on alignment quality.
- **Resampling**: Low-weight particles are discarded, while high-weight particles are duplicated to focus computation on the most probable trajectories.

Through repeated execution of these steps, gmapping simultaneously refines the robot pose estimate and incrementally builds a consistent occupancy grid map.

D. Sensor Configuration and SLAM Inputs

The SLAM algorithm was configured to use a single front-mounted laser scanner together with wheel odometry. The front-facing laser provides strong scan overlap in the direction of motion, improving scan matching performance, while odometry serves as a motion prior between successive scans. Only the front laser data were used to maintain consistency with project requirements and ensure stable map generation.

IV. Algorithm Implementation Details

The `gmapping` algorithm was implemented in an offline SLAM configuration using recorded sensor data and simulated time to ensure correct temporal synchronization. The implementation focused on algorithm-level configuration and execution, distinct from system-level data handling and visualization.

The SLAM algorithm was configured to subscribe exclusively to the `/front_scan` topic as the primary measurement input. Wheel odometry data, published on the `/odom` topic, was used internally by the Rao–Blackwellized Particle Filter (RBPF) to propagate particle states between consecutive laser scan updates. This configuration reflects a standard front-laser-based SLAM setup and ensures stable scan overlap in the direction of robot motion.

Offline execution was performed using ROS bag playback with `/use_sim_time` enabled, allowing the algorithm to process sensor data in a time-consistent and repeatable manner. During playback, the `gmapping` node incrementally estimated the robot pose and updated the occupancy grid map based on scan matching results and particle filter resampling.

Two algorithm configurations were evaluated to assess performance sensitivity to parameter selection. The first configuration used the default `gmapping` parameters to establish a baseline mapping result. The second configuration employed manually tuned parameters targeting improved scan matching accuracy, reduced ghosting, and enhanced structural consistency in the generated map. Apart from parameter adjustments, the algorithm structure, sensor inputs, and data source remained identical across both runs, ensuring a fair and controlled comparison of algorithm performance.

V. Dataset Chosen

The dataset used in this project is taken from the Rawseeds Project, specifically the benchmark problem "Laser SLAM – Bicocca 2009-02-25b." This dataset contains real-world sensor data collected from a mobile robot operating in an indoor environment at the University of Milano–Bicocca. It is widely used as a benchmark for evaluating two-dimensional laser-based SLAM algorithms and provides synchronized laser range measurements and odometry suitable for offline SLAM experiments.

The following CSV files were provided as part of the dataset:

- **front.csv**: Front-mounted two-dimensional laser scanner measurements.
- **rear.csv**: Rear-mounted two-dimensional laser scanner measurements.
- **odom.csv**: Robot odometry data describing translational and rotational motion.
- **imu.csv**: Inertial measurement unit (IMU) sensor data.

A. *Data Selection for Mapping*

Although multiple sensor streams were available, only a subset of the provided data was used for map generation in order to strictly follow the project requirements.

For the SLAM implementation in this project:

- The **front.csv** file was selected as the sole laser scan input to the SLAM algorithm.
- The **odom.csv** file was used to supply odometry information for motion prediction and pose estimation support within the gmapping framework.

The remaining data sources, **rear.csv** and **imu.csv**, were not used in the SLAM pipeline. These files were intentionally excluded to ensure that mapping was performed using only the front laser scanner, as specified in the project instructions.

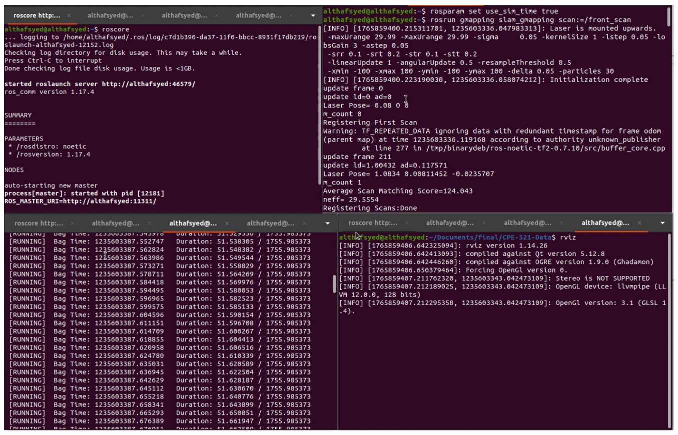
B. *Justification of Data Choice*

The combination of front laser scan data and odometry provides sufficient information for two-dimensional indoor SLAM using the gmapping algorithm. Limiting the SLAM input to a single laser scanner simplifies scan alignment, avoids ambiguity in sensor fusion, and allows a clear evaluation of mapping performance based solely on the front-mounted sensor.

VI. SLAM System Pipeline and Integration

A. *Overall System Flow*

The SLAM system follows an offline processing pipeline: **CSV** $\rightarrow$ **ROS bag** $\rightarrow$ **GMapping** $\rightarrow$ **RViz** $\rightarrow$ **Map Saving**.



```
althafsyed@althafsyed:~$ roscore
... logging to /home/althafsyed/.ros/log/c7d1b390-da37-11f0-bbcc-8931f17db219/ro
slaunch-althafsyed-12152.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://althafsyed:46579/
ros_comm version 1.17.4


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.17.4

NODES

auto-starting new master
process[master]: started with pid [12181]
ROS_MASTER_URI=http://althafsyed:11311/
```

```
althafsyed@althafsyed:~$ rosparam set use_sim_time true
althafsyed@althafsyed:~$ rosrun gmapping slam_gmapping scan:=/front_scan
[INFO] [1765859400.215311701, 1235603336.047983313]: Laser is mounted upwards.
 -maxUrange 29.99 -maxUrange 29.99 -sigma     0.05 -kernelSize 1 -lstep 0.05 -lo
bsGain 3 -astep 0.05
 -srr 0.1 -srt 0.2 -str 0.1 -stt 0.2
 -linearUpdate 1 -angularUpdate 0.5 -resampleThreshold 0.5
 -xmin -100 -xmax 100 -ymin -100 -ymax 100 -delta 0.05 -particles 30
[INFO] [1765859400.223190030, 1235603336.058074212]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.08 0 0
m_count 0
Registering First Scan
Warning: TF_REPEATED_DATA ignoring data with redundant timestamp for frame odom
(parent map) at time 1235603336.119168 according to authority unknown_publisher
         at line 277 in /tmp/binarydeb/ros-noetic-tf2-0.7.10/src/buffer_core.cpp
update frame 211
update ld=1.00432 ad=0.117571
Laser Pose= 1.0834 0.00811452 -0.0235707
m_count 1
Average Scan Matching Score=124.043
neff= 29.5554
Registering Scans:Done
```

```
 [RUNNING]  Bag Time: 1235603387.552747   Duration: 51.538305 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.562824   Duration: 51.548382 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.563986   Duration: 51.549544 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.573271   Duration: 51.558829 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.578711   Duration: 51.564269 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.584418   Duration: 51.569976 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.594495   Duration: 51.580053 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.596965   Duration: 51.582523 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.599575   Duration: 51.585133 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.604596   Duration: 51.590154 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.611151   Duration: 51.596708 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.614709   Duration: 51.600267 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.618855   Duration: 51.604413 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.620958   Duration: 51.606516 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.624780   Duration: 51.610339 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.635031   Duration: 51.620589 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.636945   Duration: 51.622504 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.642629   Duration: 51.628187 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.645112   Duration: 51.630670 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.655218   Duration: 51.640776 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.658341   Duration: 51.643899 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.665293   Duration: 51.650851 / 1755.985373
 [RUNNING]  Bag Time: 1235603387.676389   Duration: 51.661947 / 1755.985373
```

```
althafsyed@althafsyed:~/Documents/final/CPE-521-Data$ rviz
[INFO] [1765859406.642325094]: rviz version 1.14.26
[INFO] [1765859406.642413093]: compiled against Qt version 5.12.8
[INFO] [1765859406.642446260]: compiled against OGRE version 1.9.0 (Ghadamon)
[INFO] [1765859406.650379464]: Forcing OpenGl version 0.
[INFO] [1765859407.211762320, 1235603343.042473109]: Stereo is NOT SUPPORTED
[INFO] [1765859407.212189025, 1235603343.042473109]: OpenGL device: llvmpipe (LL
VM 12.0.0, 128 bits)
[INFO] [1765859407.212295358, 1235603343.042473109]: OpenGl version: 3.1 (GLSL 1
.4).
```

Fig. 1. Terminal-level execution of the ROS-based SLAM pipeline, showing ROS master startup, gmapping node execution, offline rosbag playback, and RViz visualization.

The execution sequence is summarized as follows:

- **Terminal 1 – ROS Master Initialization:** The ROS master was started using `roscore` to enable communication between all ROS nodes and manage system-wide parameters.
- **Terminal 2 – SLAM Algorithm Execution:** Simulated time was enabled to ensure synchronization with recorded data, and the `slam_gmapping` node was launched. The node was configured to subscribe exclusively to the front-mounted laser scan topic for map generation.
- **Terminal 3 – Offline Data Playback:** The recorded ROS bag file containing front laser and odometry data was replayed using `rosbag play --clock`. This provided time-consistent sensor data to the SLAM algorithm during offline execution.
- **Terminal 4 – Visualization:** RViz was launched to visualize the SLAM process in real time, including laser scans, coordinate frame transformations, robot motion, and the evolving occupancy grid map.

B. *Input Data Preparation*

The input dataset consists of recorded sensor measurements provided in CSV format, including front-mounted laser scan data and wheel odometry. A Python-based conversion script was used to convert the CSV files into standard ROS message types (`sensor_msgs/LaserScan` and `nav_msgs/Odometry`), which were stores in the form of output.bag.

C. *Offline SLAM Execution*

The generated ROS bag file was replayed using `rosbag play` with simulated time enabled. During playback, the `slam_gmapping` node subscribed to the `/front_scan` and `/odom` topics to estimate the robot pose and incrementally build the occupancy grid map.

D. *Coordinate Frame Transformations (TF)*

The SLAM system uses a hierarchical TF structure consisting of the `map`, `odom`, `base_link`, and `front_laser` frames.

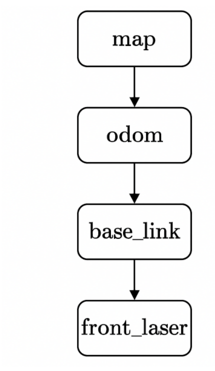
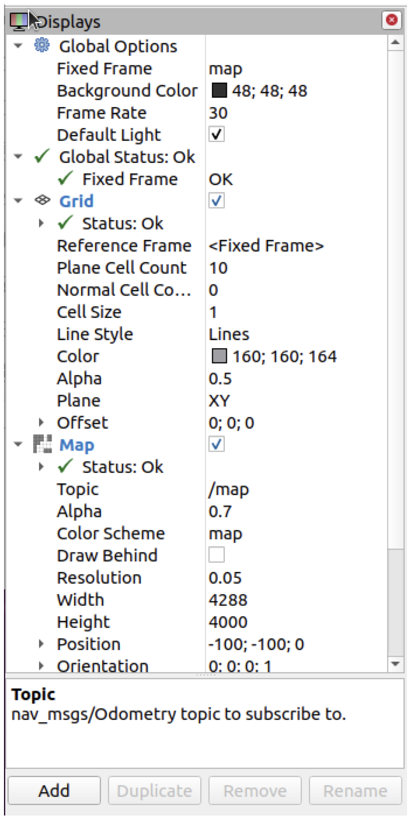


Fig. 2. ROS TF transformation tree observed during SLAM execution. The tree confirms correct connectivity between the global `map` frame, the odometry frame `odom`, the robot base frame `base_link`, and the front laser sensor frame `front_laser`.

Gmapping computes the transformation between the `map` and `odom` frames, while odometry and laser measurements are propagated through `base_link` to ensure consistent alignment in the global map frame.

VII. GUI Interface

The graphical user interface for this project was implemented using *RViz*, which was used to visualize, monitor, and validate the SLAM pipeline during offline execution. RViz provided real-time visual feedback on sensor alignment, coordinate frame consistency, and map generation while replaying recorded data. The fixed reference frame in RViz was set to `map` to ensure that all visualized data were referenced to the global map frame. Multiple RViz display modules were enabled to support validation of the SLAM process, including the occupancy grid map, laser scan data, coordinate transforms, and odometry information.



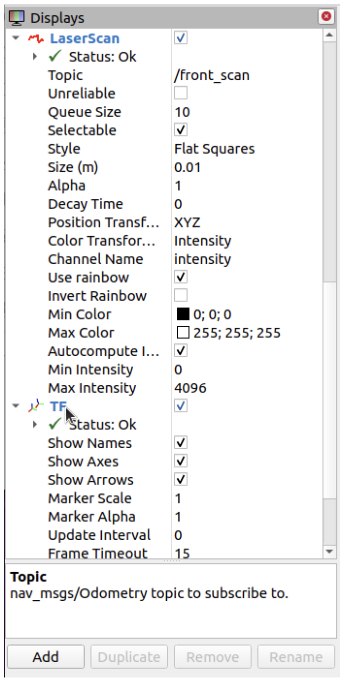
Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	10
Normal Cell Co...	0
Cell Size	1
Line Style	Lines
Color	■ 160; 160; 164
Alpha	0.5
Plane	XY
Offset	0; 0; 0
Map	✓
✓ Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.05
Width	4288
Height	4000
Position	-100; -100; 0
Orientation	0; 0; 0; 1

Topic
nav_msgs/Odometry topic to subscribe to.

Fig. 3. RViz display configuration used during SLAM execution. The fixed frame is set to `map`, and the occupancy grid map is visualized from the `/map` topic.

The `Map` display was used to visualize the occupancy grid generated by the gmapping algorithm and to monitor the incremental construction of the environment map. The `LaserScan` display, subscribed to the `/front_scan` topic, allowed verification of laser scan alignment with the map and confirmed correct scan matching behavior. The `TF` display was enabled to inspect coordinate frame relationships and ensure proper transformations between the robot frame, laser frame, and map frame. Additionally, the `Odometry` display was used to observe the robot motion trajectory derived from recorded odometry data.

RViz was used as a validation tool prior to saving the final maps. Only after confirming stable frame transformations, consistent laser alignment, and coherent map structure were the occupancy grid maps saved using the ROS map server. This ensured that the exported PGM and YAML files accurately represented the mapped environment.



LaserScan	✓
✓ Status: Ok	
Topic	/front_scan
Unreliable	☐
Queue Size	10
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transf...	XYZ
Color Transfor...	Intensity
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0
Max Color	□ 255; 255; 255
Autocompute I...	✓
Min Intensity	0
Max Intensity	4096
TF	✓
✓ Status: Ok	
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Marker Alpha	1
Update Interval	0
Frame Timeout	15

Topic
nav_msgs/Odometry topic to subscribe to.

Fig. 4. RViz visualization of the front laser scan (`/front_scan`) with TF enabled to validate frame connectivity during mapping.

VIII. Map Saving and Repeatability

After validating stable scan alignment and frame connectivity in RViz, the final occupancy grid map was exported using the ROS `map_server` package. The `map_saver` utility saved the map in the standard `.pgm` (image) and `.yaml` (metadata) formats, which were later used for reporting and comparison between runs.

To ensure repeatability, both the baseline and tuned evaluations were executed using the same input bag file (total playback duration $\approx$ 1755 s) with `/use_sim_time` enabled. Only the `slam_gmapping` parameter set was changed between runs; the sensor topics, data source, and playback procedure remained identical. This controlled setup ensures that differences in map quality are attributable to parameter tuning rather than changes in input data or timing.

IX. Simulation Results and Performance Discussion

This section presents the results obtained from offline SLAM simulations using the `slam_gmapping` algorithm. Two configurations were evaluated: a baseline setup using default parameters and a refined setup using tuned parameters. The performance of each configuration was assessed qualitatively based on map clarity, structural consistency, and scan alignment.

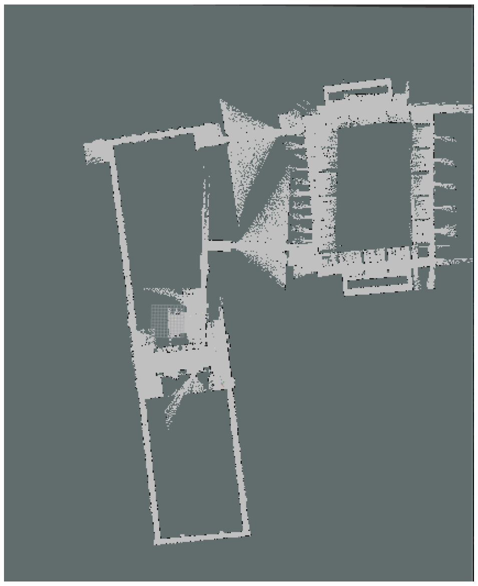


Fig. 5. Initial SLAM output during offline execution using front laser data prior to parameter tuning, illustrating scan fan artifacts and pose uncertainty.

Figure 5 provides an intermediate view of the SLAM process during offline execution, capturing the mapping behavior before any parameter optimization was applied. This result highlights the sensitivity of particle-filter-based SLAM to scan matching quality and motion uncertainty when operating with default assumptions. The observed inconsistencies emphasize the need for careful parameter selection in `gmapping` to balance pose estimation stability and map fidelity, motivating the comparative evaluation between default and tuned configurations discussed in the following sections.

A. Baseline Simulation (Default Parameters)

The baseline SLAM experiment was performed using the default parameter values provided by the `slam_gmapping` package. This configuration serves as a reference to evaluate the effectiveness of parameter tuning.

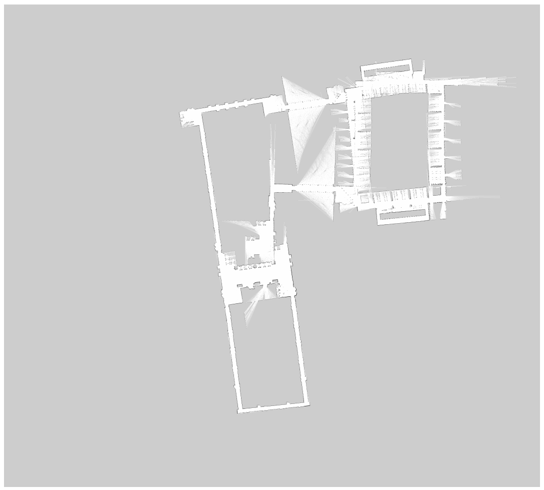


Fig. 6. Occupancy grid map generated using default `gmapping` parameters. The overall environment structure is visible, but mapping artifacts are present.

As shown in Fig. 6, the baseline map successfully captures the global layout of the environment, including corridors and room boundaries. However, several limitations are observed. In particular, wall structures in long corridors appear duplicated or blurred, indicating scan-matching errors. Additionally, noise and fuzzy occupied cells are visible in open areas, which suggests insufficient filtering of low-quality pose hypotheses during mapping.

B. Refined Simulation (Tuned Parameters)

To improve mapping accuracy, key `gmapping` parameters related to scan matching frequency and particle acceptance were refined.

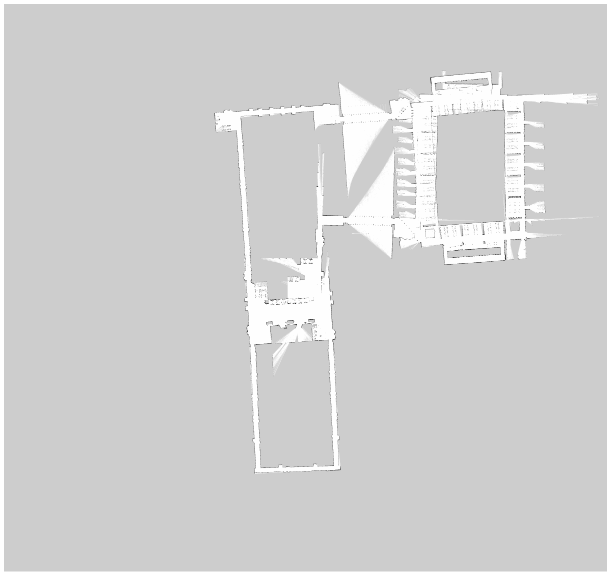


Fig. 7. Occupancy grid map generated using refined `gmapping` parameters. Improved wall definition and reduced noise are observed.

The tuning strategy focused on enforcing stricter scan alignment and reducing the influence of poor pose estimates.

The resulting map, shown in Fig. 7, demonstrates a noticeable improvement in structural clarity. Wall boundaries appear sharper and more consistent, and ghosting artifacts are significantly reduced. Open areas exhibit less scattered occupancy, indicating improved pose convergence and map stability.

C. Tuned `gmapping` Parameter Changes

The tuned SLAM run used the same `output.bag` input data as the baseline configuration; only the `slam_gmapping` parameters were modified to improve scan matching stability and reduce mapping artifacts. Parameter tuning focused on increasing update frequency, enforcing stricter scan matching acceptance, and improving particle filter robustness.

TABLE I
KEY `GMAPPING` PARAMETER CHANGES BETWEEN BASELINE AND TUNED CONFIGURATIONS.

Parameter	Default	Tuned	Effect
`linearUpdate`	1.0	0.5	More frequent scan updates with smaller translations
`angularUpdate`	0.5	0.2	More frequent updates for small rotations
`minimumScore`	0	150	Rejects weak scan matches, reducing ghosting
`maxUrange`	-1	8.0	Limits unreliable long-range laser beams
`particles`	30	50	Improves robustness of RBPF pose hypotheses

Reducing the linear and angular update thresholds increased scan matching frequency, limiting the accumulation of pose error. Increasing the minimum score requirement prevented poor scan alignments from being accepted, significantly reducing double-wall artifacts. Limiting the maximum usable laser range reduced noise from distant measurements, while increasing the particle count improved stability in ambiguous regions.

In the tuned configuration, scan updates were triggered more frequently (via `linearUpdate` and `angularUpdate`) and scan matching was made stricter (via `minimumScore`). In addition, limiting `maxUrange` reduced far-range noise, while adjusting `particles` improved stability in ambiguous regions. Collectively, these changes improved wall sharpness and reduced mapping artifacts compared to the baseline map.

D. Comparative Performance Analysis

A qualitative comparison between the baseline and refined simulations is summarized in Table II. The tuned configuration consistently outperforms the baseline setup across all evaluated criteria.

TABLE II
COMPARISON OF SLAM PERFORMANCE USING DEFAULT AND TUNED PARAMETERS

S. No	Metric	Default	Tuned
1	Wall definition	Blurred / duplicated	Sharp and consistent
2	Mapping noise	High in open areas	Significantly reduced
3	Scan alignment	Inconsistent	Stable and accurate
4	Structural consistency	Moderate	High
5	Overall map quality	Acceptable	Improved

The observed improvements can be attributed to the behavior of the Rao–Blackwellized Particle Filter underlying the `gmapping` algorithm. Stricter scan acceptance criteria and more frequent scan matching result in better particle convergence, reducing accumulated pose error and producing a more coherent occupancy grid map.

Overall, these results demonstrate that parameter tuning plays a critical role in enhancing SLAM performance, even when using identical sensor data and system configuration.

- Note: Both the baseline and tuned SLAM simulations were executed using the same ROS bag file, corresponding to a total playback duration of approximately 1755 seconds, ensuring a fair and consistent comparison between configurations.

X. DISCUSSION AND REFLECTION

This project provided practical insight into the challenges involved in implementing a reliable SLAM system using real-world logged sensor data. Beyond algorithm selection, the work highlighted the importance of system integration, parameter tuning, and validation when deploying SLAM within a ROS-based framework.

A. Technical Challenges and Lessons Learned

Coordinate Frame Integrity (TF Tree): A key lesson learned was the critical role of a consistent Transformation (TF) tree. Initial execution attempts produced frame connectivity errors, demonstrating that successful SLAM depends not only on the mapping algorithm but also on correct publication of static transformations (e.g., `/base_link` → `/front_laser`) and dynamic transformations (e.g., `/odom` → `/base_link`). Proper conversion of the CSV data into a time-synchronized ROS bag was essential to maintaining temporal consistency and TF integrity during offline execution.

RBPF Parameter Sensitivity: The comparison between baseline and tuned results revealed the sensitivity of the Rao–Blackwellized Particle Filter (RBPF) used by the `gmapping` algorithm. Parameters such as `minimumScore`, `linearUpdate`, and `angularUpdate` strongly influenced scan acceptance and pose correction. Conservative settings led to incomplete mapping, while overly permissive settings caused ghosting and duplicated structures. Effective tuning required balancing these parameters based on the characteristics of the Rawseeds dataset.

Offline Processing Methodology: Using `rosbag` playback with simulated time (`/use_sim_time`) enabled repeatable and fair evaluation of different configurations. This approach ensured that both baseline and tuned runs were executed under identical conditions, improving reproducibility and confidence in the reported results.

B. Limitations and Future Work

Single-Sensor Configuration: The SLAM pipeline relied solely on a front-mounted laser scanner and wheel odometry, consistent with project requirements. While effective for indoor mapping, this configuration remains susceptible to odometry drift, particularly in extended or feature-poor environments.

Future Sensor Fusion: Incorporating inertial data from the `imu.csv` dataset using sensor fusion methods such as an Extended Kalman Filter (EKF) could improve pose estimation accuracy and further reduce mapping artifacts.

Algorithmic Extensions: Although `gmapping` performs well for 2D laser-based SLAM, its RBPF-based approach may not scale optimally to larger environments. Future work could explore optimization-based SLAM systems such as Cartographer to compare mapping accuracy and computational performance.

XI. Conclusion

This project implemented a complete two-dimensional SLAM pipeline using the `gmapping` algorithm within the ROS Noetic framework. Offline processing of real-world sensor data demonstrated that accurate environment mapping can be achieved using only a front-mounted laser scanner and odometry. Comparative results showed that parameter tuning significantly improves map clarity and structural consistency over the default configuration. Overall, the project highlights the importance of proper system integration and parameter selection for reliable SLAM performance.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[2] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao–Blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[3] L. Iocchi, D. Nardi, G. Grisetti, and C. Stachniss, “Rawseeds: Robotics advancing intelligence through simulation,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009. Dataset: Laser SLAM – Bicocca 2009-02-25b.

[4] Canonical Ltd., “Ubuntu Linux Operating System,” accessed 2024. [Online]. Available: https://ubuntu.com

[5] UTM Virtual Machines for macOS, “UTM Documentation,” accessed 2024. [Online]. Available: https://mac.getutm.app

[6] Open Source Robotics Foundation, “ROS Noetic Ninjemys,” 2020. [Online]. Available: https://wiki.ros.org/noetic

[7] Open Source Robotics Foundation, “slam_gmapping Package,” 2020. [Online]. Available: https://wiki.ros.org/slam_gmapping