

Probabilistic SLAM with Autonomous Exploration in Simulation

Ramy Mekhzer, Bassel Kenaan, Sahib Singh

Abstract—This project explores Simultaneous Localisation and Mapping (SLAM) and autonomous exploration using a simulated robot in Webots. We built a particle-filter based SLAM system (inspired by FastSLAM) that uses noisy wheel odometry and LiDAR data to estimate where the robot is while building a map of the environment. We used custom models to handle sensor noise and update the robot’s belief about the world.

The main focus of the work is exploration. Instead of just moving randomly, the robot uses “frontier-based exploration” to find the edges of the known map and move towards areas it hasn’t seen yet. We compared this smart strategy against a simple random movement strategy to see which one works better.

We tested both strategies in different simulated worlds. Our results show that while directed exploration is generally faster and more efficient, it can struggle in narrow corridors where random movement is sometimes more robust.

I. INTRODUCTION¹

A. Context and Motivation

Autonomous mobile robotics increasingly underpins modern automation, from domestic service robots to large-scale industrial and search-and-rescue applications. A fundamental capability in such systems is operating in previously unseen environments without access to external positioning such as GPS. This requires solving *Simultaneous Localisation and Mapping* (SLAM), in which a robot estimates both its pose and a map of the environment, together with *Autonomous Exploration*, which determines where the robot should move to efficiently gather new information. Robust SLAM and exploration pipelines are critical for tasks such as warehouse logistics, environmental monitoring, and inspection of hazardous or inaccessible regions.

B. Problem Statement

This project focuses on enabling a differential-drive mobile robot to autonomously map an unknown 2D environment using only local sensing. The robot must cope with noisy wheel odometry and limited-range LiDAR observations, both of which introduce uncertainty into localisation and mapping. There are two main challenges: (i) maintaining a consistent estimate of the robot’s pose and the occupancy-grid map, and (ii) selecting motion commands that maximise information gain while avoiding the inefficiencies of uninformed or random movement.

C. Approach Overview

We develop a simulation-based framework in Webots for investigating probabilistic SLAM and exploration. The system implements a particle-filter SLAM algorithm inspired by FastSLAM [1], [2], representing the posterior over robot poses through a set of weighted samples that naturally capture non-Gaussian uncertainty. Mapping is performed on an occupancy grid updated via a custom step-window inverse sensor model [3] tailored to the behaviour of the simulated LiDAR. To enable autonomous exploration, we implement a frontier-based strategy [4] that selects navigation targets on the boundary between known free space and unexplored regions, and compare it against a baseline random-exploration strategy.

We also acknowledge the open-source contributions from PythonRobotics [5] which provided reference implementations for motion models and particle filter structures.

D. Contributions

The main contributions of this work are:

- 1) We implemented a particle-filter SLAM system from scratch in Python, without using pre-built SLAM libraries.
- 2) We developed a “step-window” inverse sensor model to improve how walls are drawn on the map and reduce noise.
- 3) We built a modular framework to compare different exploration strategies, specifically Frontier-based vs Random.
- 4) We analysed the performance of these strategies using metrics like map coverage and entropy (see Table I and Table II).

E. Structure of the Paper

The remainder of this report is structured as follows. Section II reviews related work on particle-filter SLAM, inverse sensor models, and exploration algorithms. Section III describes the methodology, including the motion and measurement models, mapping process, and exploration strategies. Section IV outlines the experimental setup and presents the results. Section V discusses limitations of the system and potential directions for future development.

II. RELATED WORK²

Intelligent Robotics 06-30227, School of Computer Science, University of Birmingham, UK.

¹Section Author: Sahib Singh

²Section Author: Ramy Mekhzer

Autonomous navigation in unknown environments draws on three tightly coupled research areas: probabilistic localisation, environmental mapping, and exploration strategies. This section summarises key developments relevant to the present work.

A. Probabilistic SLAM and Particle Filters

Simultaneous Localisation and Mapping (SLAM) addresses the fundamental dependency between localisation and mapping, where neither can be solved independently. Early solutions such as EKF-SLAM modelled uncertainty through a Gaussian distribution and jointly estimated the robot pose and landmark positions. Although effective in small-scale settings, EKF-SLAM is sensitive to non-linear motion models and scales poorly due to its quadratic complexity.

Particle-filter approaches were introduced to overcome these limitations. Algorithms such as FastSLAM factorise the posterior into a distribution over robot trajectories and a set of conditionally independent landmark estimates. Each particle represents a hypothesis of the robot's pose, and importance resampling concentrates computational effort on the most plausible hypotheses. Particle-filter SLAM is robust to multi-modal belief distributions and accommodates non-Gaussian noise, making it well suited to occupancy-grid mapping in 2D environments. It also naturally handles failure cases such as the "kidnaped robot" scenario.

B. Occupancy Grid Mapping and Inverse Sensor Models

Occupancy-grid mapping discretises the environment into a 2D grid where each cell stores the probability of being occupied. This representation explicitly models free, occupied, and unknown space, making it widely used in navigation and path planning. Bayesian binary filtering, often implemented using log-odds updates, enables the fusion of successive sensor readings while maintaining numerical stability.

A key component of grid-based mapping is the inverse sensor model, which converts raw LiDAR range measurements into occupancy updates. Simple endpoint-only models tend to produce sparse or noisy maps, especially in cluttered environments. More expressive models, such as "thick-wall" or beam-based formulations, update not only the endpoint but also the free space along each ray, and can better capture uncertainty in the sensor. These models produce sharper obstacle boundaries and reduce artefacts, which is critical for downstream tasks such as frontier extraction.

C. Frontier-Based Exploration³

With a functional SLAM pipeline, autonomous exploration determines how the robot should move to efficiently reduce map uncertainty. Frontier-based exploration, introduced by Yamauchi, identifies *frontiers* (the boundaries between known free space and unexplored regions) as natural targets for information gain. By repeatedly navigating to these frontiers, the robot expands its mapped area systematically.

Frontier methods contrast with stochastic strategies such as random walk, which require minimal computation and are robust but generally inefficient for large-scale coverage. Frontier-based exploration leverages spatial structure to guide the robot, but its performance is sensitive to the accuracy and resolution of the underlying occupancy grid. Errors in localisation or mapping may cause unreachable or misidentified frontiers, leading to failure cases such as oscillation or premature termination.

D. Summary

This work builds directly on the established foundations of particle-filter SLAM, occupancy-grid mapping, and frontier-based exploration. Rather than introducing new SLAM algorithms, our focus lies in implementing these components end-to-end within the Webots simulation environment and analysing their interaction. In particular, we evaluate how map quality (affected by noise, particle degeneracy, and inverse sensor modelling) influences the relative performance of frontier-driven and random exploration strategies.

III. METHODS

A. Robot Platform and Simulation Environment⁴

The experimental validation is conducted within the Webots simulation environment, utilising a custom differential drive robot model encapsulated in the `MyRobot` class. The robot's kinematic configuration consists of two independent DC motors driving wheels with a radius of 0.125 m and a separation of 0.45 m. Environmental perception is achieved through a multi-modal sensor suite, primarily relying on a 360-degree LiDAR sensor for high-fidelity range measurements up to 4 meters, and wheel encoders that provide feedback for odometry estimation. The simulation operates on a fixed control cycle with a timestep of 64 ms, ensuring consistent temporal resolution for both control and sensing operations.

B. SLAM Framework Overview⁵

The Simultaneous Localization and Mapping (SLAM) system is implemented using a particle filter approach derived from the FastSLAM algorithm [1]. To balance computational efficiency with robust localisation, the system employs a "best particle" mapping strategy. While this sacrifices the theoretical consistency of maintaining maps per particle (particularly during loop closure), it significantly reduces memory and computational overhead for our real-time Python implementation.

Particle diversity is maintained through a low-variance resampling mechanism, which is triggered dynamically when the effective sample size

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^N w_i^2}$$

falls below 50% of the total number of particles, i.e., $N_{\text{eff}} < 0.5N$.

³Subsection Author: Bassel Kenaan

⁴Subsection Author: Sahib Singh

⁵Subsection Author: Ramy Mekhzar

C. Motion and Measurement Models⁶

The probabilistic motion model assumes differential drive kinematics with additive Gaussian noise [5]. Given commanded linear and angular velocities v and ω , we model the perturbed controls as

$$\begin{aligned} v_t &= v + \epsilon_v, & \epsilon_v &\sim \mathcal{N}(0, \sigma_v^2), \\ \omega_t &= \omega + \epsilon_\omega, & \epsilon_\omega &\sim \mathcal{N}(0, \sigma_\omega^2). \end{aligned}$$

For a time step Δt , the pose of each particle is propagated according to

$$\begin{aligned} x_{t+1} &= x_t + v_t \Delta t \cos(\theta_t), \\ y_{t+1} &= y_t + v_t \Delta t \sin(\theta_t), \\ \theta_{t+1} &= \theta_t + \omega_t \Delta t. \end{aligned}$$

The measurement model evaluates the likelihood of a particle by comparing its predicted LiDAR scan to the global occupancy grid. For each beam i with measured range z_i and bearing ϕ_i , the endpoint under the particle's pose hypothesis (x, y, θ) is computed as

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + z_i \begin{pmatrix} \cos(\theta + \phi_i) \\ \sin(\theta + \phi_i) \end{pmatrix}.$$

Let $m(x_i, y_i)$ denote the occupancy probability of the grid cell containing the projected endpoint. The likelihood of the full scan is then

$$w = \prod_{i=1}^{N_{\text{beams}}} p(m(x_i, y_i) | z_i),$$

or equivalently, using log-likelihood for numerical stability,

$$\log w = \sum_{i=1}^{N_{\text{beams}}} \log p(m(x_i, y_i) | z_i).$$

We model $p(m(x_i, y_i) | z_i)$ using an endpoint scoring function: endpoints that fall on occupied cells increase likelihood, while endpoints in free space decrease it, penalising particles whose predicted measurements contradict the current map.

D. Step-Window Inverse Sensor Model⁷

A critical component of grid-based mapping is translating 1D range measurements into 2D map updates. Standard Bresenham ray-casting often produces thin, disconnected walls if only the exact endpoint is marked. To improve map consistency and ensure solid obstacles, we implemented a custom *Step-Window* inverse sensor model [3].

The algorithm proceeds as follows for each valid LiDAR beam:

- 1) **Ray Tracing:** We compute the integer grid cells along the ray from the robot's position to the measured endpoint using Bresenham's line algorithm.
- 2) **Thick-Wall Heuristic:** Instead of marking only the single terminal cell as occupied, we mark the *last two cells* of the ray as occupied. This creates a "thickened" wall

effect that is more robust to slight particle inaccuracies and sensor noise.

- 3) **Free Space Clearing:** All cells preceding the last two are updated with a negative log-odds value (mark free), effectively carving out empty space.
- 4) **Sticky Walls:** To prevent noise from eroding static walls, we employ a conservative rule: if a cell's log-odds $l(c)$ exceeds a threshold $l_{\text{occ-th}}$, we do not apply free-space updates.

The log-odds update for a cell c_{ij} is given by:

$$l(c_{ij}) \leftarrow \text{clamp}(l(c_{ij}) + \delta, l_{\min}, l_{\max})$$

where $\delta = \delta_{\text{occ}}$ for the last two cells, and $\delta = \delta_{\text{free}}$ otherwise.

E. Frontier-Based Exploration⁸

Frontier-based exploration provides the robot with a structured strategy for selecting informative motion targets [4]. A frontier is defined as the boundary between free space and unknown space. Moving towards these boundaries ensures the robot prioritises regions where new information is available.

Our extraction process operates on the ternary classification of the log-odds grid:

- **Free:** $l(c) < l_{\text{free-th}}$
- **Occupied:** $l(c) > l_{\text{occ-th}}$
- **Unknown:** Otherwise

A cell is a frontier point if it is Free and has at least one Unknown neighbour. Frontier points are clustered using Euclidean distance, and the centroid of a cluster becomes a potential navigation target. The robot selects the nearest valid frontier centroid and navigates using an A* path planner [5].

F. Random Exploration Baseline⁹

As a baseline, we implemented a Random Exploration strategy. The robot defaults to moving forward until an obstacle is detected within a critical threshold (e.g., 0.8m). Upon detection, the robot stops and executes a random turn (left or right) for a fixed duration before resuming forward motion. This "bounce-off-walls" behaviour requires no global map maintenance for navigation, serving as a lower bound for exploration efficiency.

IV. EXPERIMENTAL SETUP¹⁰

Experiments were conducted in the Webots simulator across four distinct environments to evaluate robustness:

- 1) `slam_project.wbt`: A baseline environment representing an average-sized house with four rooms.
- 2) `large_apartment.wbt`: A larger residential layout featuring long corridors and some narrow halls.
- 3) `maze.wbt`: A complex environment constrained by very narrow hallways.
- 4) `small_room.wbt`: A single small room serving as a minimal test case.

⁶Subsection Author: Ramy Mekhzer

⁸Subsection Author: Bassel Kenaan

⁹Subsection Author: Bassel Kenaan

¹⁰Section Author: Bassel Kenaan

⁷Subsection Author: Ramy Mekhzer

TABLE I
BENCHMARK RESULTS (1000 STEPS) - BASELINE WORLD

Metric	Frontier	Random
Final Coverage	60.83%	34.04%
Map Entropy (Uncertainty)	0.848	0.915
Explored Area (m^2)	15.2	8.5

For all tests, the occupancy grid was initialised with a size of 400×400 cells and a resolution of 0.05 m (5 cm), covering a 20×20 m area.

We evaluated performance using the following metrics:

- 1) **Coverage Percentage:** The ratio of explored cells (free or occupied) to the total navigable area of the map (determined by flood-fill from the start position).
- 2) **Entropy:** A proxy for map uncertainty, computed as

$$H = \frac{1}{|C|} \sum_{c \in C} [-p_c \log p_c - (1 - p_c) \log(1 - p_c)],$$

where p_c is clamped to $[\epsilon, 1 - \epsilon]$ (with $\epsilon = 10^{-6}$) to avoid $\log(0)$.

- 3) **Completion Time:** Time steps required to reach a steady-state coverage (see Table I for results).

V. RESULTS¹¹

A. Comparison of Strategies

Performance varied significantly across the four test environments.

In the `slam_project.wbt` (baseline) and `small_room.wbt`, the Frontier-based strategy demonstrated superior efficiency, systematically exploring rooms and achieving high coverage. The structured nature of the environment allowed the frontier detector to identify clear targets.

However, in the `large_apartment.wbt`, the performance trend reversed. We performed a controlled benchmark in the baseline environment (`slam_project.wbt`) to quantify the efficiency gap between the two strategies. Both agents were run for a fixed duration of 1000 time steps over 10 independent trials. The results (averaged), summarised in Table I, clearly demonstrate the advantage of structured exploration in this environment.

In this structured environment with clear rooms and corridors, the Frontier-based strategy avoided the redundancy inherent in the random walk approach. It achieved nearly double the coverage (60.8% vs 34.0%) and significantly lower map entropy (0.848 vs 0.915), confirming that directed exploration leads to much faster convergence when the environment structure supports accurate frontier detection.

B. Quantitative Analysis: Large Environment

We repeated the benchmark in the larger, more complex environment (`large_apartment.wbt`), averaging performance over 10 runs. The results, shown in Table II, reveal a reversal in performance.

TABLE II
BENCHMARK RESULTS (1000 STEPS) - LARGE APARTMENT

Metric	Frontier	Random
Final Coverage	56.81%	62.23%
Map Entropy	0.858	0.844
Explored Area (m^2)	14.2	15.6

In this setting, the Random agent outperformed the Frontier agent (62.2% vs 56.8% coverage). This result highlights a critical limitation: reliance on map accuracy. In narrow corridors, slight sensor noise caused the SLAM system to register “spurious occupied cells” in the middle of hallways. The A* path planner treated them as impassable, causing the robot to oscillate. The Random agent, being reactive, simply bounced off real walls and drifted through these gaps, achieving better long-term coverage.

C. SLAM Performance

The particle filter successfully maintained localisation in static environments. The “Thick-Wall” sensor model proved effective in generating contiguous walls, reducing the likelihood of the path planner finding paths through solid obstacles. However, rotational drift remains a challenge; rapid turns often lead to “map tearing,” where a single room is duplicated at a slight angle.

VI. DISCUSSION¹²

A. Failure Modes

A common failure mode observed was **Stagnation** due to unreachable frontiers. Occasionally, a frontier would be detected behind a wall (due to sensor noise). The path planner would fail to find a path, but the frontier remained the “best” target. We addressed this by implementing a **Blacklist** mechanism: if a target is unreachable for multiple attempts, it is added to a blacklist and ignored for future selection.

Another issue is **Kidnapping**. If the robot collides with a wall and gets physically displaced without wheel encoder feedback (wheel slip), the particle filter diverts from ground truth. Recovering from this requires a more robust particle injection strategy or visual relocalisation, which was outside the scope of this project.

We also observed **Map Corruption during Rotations**. Fast spins often caused the map to “tear” (duplicate or rotated structures due to angular drift). This happens because our motion model under-estimates the angular slip, and the scan matcher gets stuck in a local maximum.

Finally, **Particle Depletion** occurred in featureless corridors. With few unique landmarks (corners), most particles had similar likelihood scores, causing the filter to spread out and lose track of the precise x-coordinate.

VII. CONCLUSION AND LIMITATIONS¹³

We successfully implemented a probabilistic SLAM system and compared Frontier-Based exploration against a Random

¹¹Section Author: Bassel Kenaan

¹²Section Author: Ramy Mekhzer

¹³Section Author: Ramy Mekhzer

baseline. While the system works well in ideal conditions, our results demonstrate that "smarter" is not always better. In complex environments with sensor noise, the robustness of a reactive random walker can outperform a fragile planning-based agent.

Limitations:

- The current implementation uses a single "best particle" map, which loses the diversity of hypotheses in the full FastSLAM approach especially during loop closure.
- The system assumes a static environment; dynamic obstacles (like walking humans) would leave "streaks" in the map.
- Computation time scales with grid interactions; ray-casting every beam on a Python-based grid is the primary bottleneck.

VIII. CODE

The full implementation of the probabilistic SLAM system, including experiment scripts, Webots simulation worlds, and evaluation code, is available at:

<https://github.com/Ramy3077/>

Probabilistic-SLAM-with-Autonomous-Exploration-in-Simulation-Uob

The repository contains instructions for reproducing the experiments reported in this paper.

REFERENCES

- [1] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *AAAI/IAAI*, vol. 593598, 2002.
- [2] M. Kaess, "Fastslam," <https://github.com/michael-kaess/fastslam>, 2015, accessed: 2025-12-12.
- [3] D. Büscher, "Occupancy grid mapping," https://github.com/srl-freiburg/occupancy_grid_mapping, 2018, accessed: 2025-12-12.
- [4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. Towards New Computational Principles for Robotics and Automation*. IEEE, 1997, pp. 146–151.
- [5] A. Sakai, "Pythonrobotics: A python code collection of robotics algorithms," <https://github.com/AtsushiSakai/PythonRobotics>, 2018.