Particle Filter SLAM and 2D Occupancy Grid Mapping

Alif Ilham Madani

April 13, 2026

1 Introduction

This project details the implementation of a 2D Occupancy Grid Mapping system using a differential-drive robot equipped with wheel encoders and a Hokuyo UTM-30LX Lidar scanner. In this first phase, the mapping relies strictly on dead reckoning (wheel odometry) to estimate the robot's trajectory. The environment is mapped by projecting Lidar scans into a global coordinate frame and updating a discretized probability grid using log-odds.

2 Problem Formulation

The objective is to reconstruct a 2D map of an unknown environment and track the robot's trajectory within it using only raw sensor data. The input data consists of synchronized timestamps, 4-channel wheel encoder counts, and 2D Lidar range/angle scans. The system must:

1. Calculate the robot's continuous pose (x_t, y_t, θ_t) in a global reference frame.
2. Transform local Lidar polar coordinates into global Cartesian coordinates.
3. Initialize and update a discrete 2D occupancy grid map using Bresenham's ray-casting algorithm to distinguish between obstacles and free space.

3 Technical Approach

3.1 Dead-Reckoning and Kinematics

The robot's movement is estimated by accumulating encoder ticks. The platform features wheels with a 127 mm radius and encoders operating at 360 counts per revolution. The encoder counts from the left wheels (FL, RL) and right wheels (FR, RR) are averaged to find the distance traveled by each side (d_L, d_R). The linear distance traveled by the center of the robot $(d_{center,t})$ and the change in heading $(\Delta\theta_t)$ at a given time step t are calculated using these side distances and the robot's effective width (W_{eff}) [1]:

$$d_{center,t} = \frac{d_R + d_L}{2}$$

$$\Delta\theta_t = \frac{d_R - d_L}{W_{eff}}$$

Using these localized changes, the robot's global heading (θ_t) and 2D global position (x_t, y_t) are iteratively updated through numerical integration (cumulative summation) at each discrete time step:

$$\theta_t = \theta_{t-1} + \Delta\theta_t$$

$$x_t = x_{t-1} + d_{center,t}\cos(\theta_t)$$

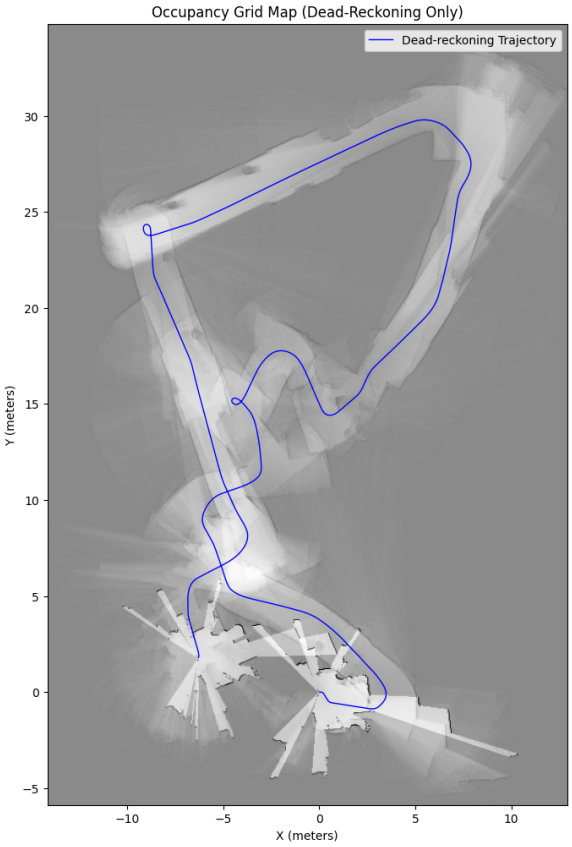
$$y_t = y_{t-1} + d_{center,t}\sin(\theta_t)$$

3.1.1 Effective Robot Width Tuning and Wheel Scrubbing

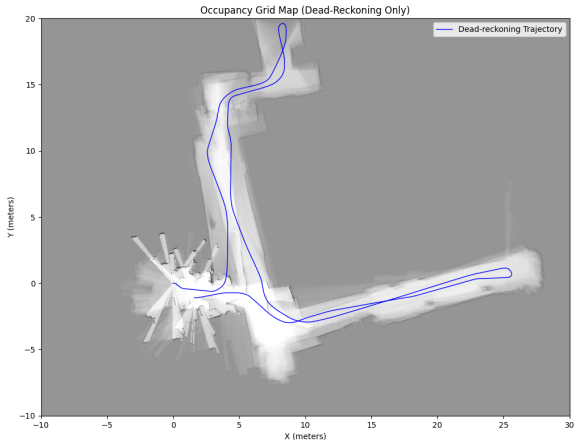
While the physical schematic indicates a robot width of 476.25 mm, using this exact physical measurement (1.0× multiplier) resulted in significant rotational errors (Figure 1a). This error is caused by a physical phenomenon known as wheel scrubbing.

Because the robot utilizes a skid-steer differential drive, the wheels cannot pivot; they must physically drag (or scrub) laterally across the floor to execute a turn. This lateral friction acts as turning resistance. Consequently, the robot requires more encoder ticks to achieve a physical rotation than the raw physical geometry suggests.

If the un-tuned physical width is used, the kinematic model assumes a perfect, frictionless pivot and thus overestimates the rotation. This injects a systematic heading drift ($\Delta\theta$ error) into the trajectory at every turn. As seen in Figure 1a, this heading drift explicitly manifests as map distortion, where straight corridors incorrectly bow and curve because the projected Lidar rays are rotated by the accumulating θ error. By applying a 1.5× multiplier to calculate an effective width ($W_{eff} = 714.375$ mm), we mathematically compensate for the energy lost to wheel scrubbing, neutralizing the systematic heading drift and restoring the straight corridors and true 90° corners seen in Figure 1b.



(a) Un-tuned (1.0× multiplier): Severe rotational drift causing corridors to curve.



(b) Tuned (1.5× multiplier): Corrected geometry showing straight corridors and 90° corners.

Figure 1: Comparison of the generated occupancy grid map for Map 20 before and after tuning the effective robot width.

3.2 Coordinate Transformations

To map the environment, the raw Lidar scans must be transformed from the sensor's local polar frame to the global world frame.

1. **Polar to Cartesian:** The scan ranges and angles are converted to local coordinates (x_{local}, y_{local}). Valid range thresholds are enforced, filtering out points below 0.1 m or above 30 m.

2. **Global Transformation:** A 2D rotation matrix utilizing the robot's global heading θ_t is applied to the local coordinates, followed by a translation utilizing the robot's global position (x_t, y_t).

3.3 Occupancy Grid Mapping

The continuous physical world is discretized into a 2D array with a resolution of 0.05 m (50 mm) per cell. To ensure the map accommodates the entire run, the grid dimensions are dynamically scaled to the minimum and maximum X and Y coordinates of the dead-reckoning trajectory, with an additional 5 meters of padding on all sides.

To maintain numerical stability and optimize memory, the grid is stored as an `int8` array, and probabilities are updated using integer approximations of the log-odds for-

mulation [1]. For each Lidar scan, the global coordinates of the hits are converted into discrete grid indices.

- **Hits (Obstacles):** The grid cells corresponding to the Lidar returns are incremented by a positive log-odds value of +2, increasing the probability of an obstacle.
- **Misses (Free Space):** Bresenham's line algorithm [2] is utilized to extract all grid cells intersecting the ray between the robot's center and the Lidar hit. These cells are decremented by a negative log-odds value of −1, representing free space.

To prevent overconfidence and avoid `int8` overflow, map values are strictly clipped within a defined probability bounds of $[-100, 100]$.

3.4 Particle Filter SLAM

To correct the accumulated drift inherent in dead-reckoning, a Particle Filter (Sequential Monte Carlo) algorithm is implemented [1]. The filter maintains a set of $N = 100$ particles, each representing a distinct hypothesis of the robot's state (x, y, θ). The algorithm operates in four primary steps: Prediction, Update, Estimation, and Resampling.

3.4.1 Prediction and True Kinematic Motion Model

At each time step, the global odometry deltas $(\Delta x, \Delta y)$ are transformed into the robot's local frame to calculate forward and lateral movements (dx_{local}, dy_{local}). This ensures that particles move kinematically relative to their own distinct headings rather than blindly translating along the global odometry path.

$$dx_{local} = \Delta x \cos(\theta_{t-1}) + \Delta y \sin(\theta_{t-1})$$
$$dy_{local} = -\Delta x \sin(\theta_{t-1}) + \Delta y \cos(\theta_{t-1})$$

To model the uncertainty in movement, proportional Gaussian noise is injected into the local translations and rotations. Unlike fixed noise, which violently scatters particles during slow, straight movements, proportional noise scales with the physical magnitude of the movement. Forward translation noise is kept tightly clamped to force the filter to trust the wheel encoders in featureless corridors, while rotational noise is increased to allow the particle cloud to fan out naturally during turns.

3.4.2 Update Step and Sensor Capping

To evaluate the likelihood of each particle's hypothetical pose, its simulated Lidar scan is projected onto the current occupancy grid. The correlation score is calculated by summing the map's log-odds values at the projected hit coordinates.

Crucially, the Lidar range is capped at 15 m. Incorporating distant Lidar returns (e.g., 30 m) introduces severe rotational drift, where a 1° heading error results in massive translational map errors at maximum range. Furthermore, capping the range mitigates longitudinal degeneracy in long corridors by filtering out max-range sensor noise. By restricting the sensor to 15 m, the filter is forced to rely on high-confidence local features to anchor the particles.

The correlation scores are normalized using a temperature constant (T) to prevent the weights from collapsing instantly, yielding the updated weight (w_t) for each particle:

$$w_t = w_{t-1} \exp\left(\frac{\text{score} - \text{max\_score}}{T}\right)$$

3.4.3 Estimation and Resampling

The robot's true trajectory is estimated using a weighted average of all particle poses, ensuring a smooth, continuous path rather than a jittery sequence of maximum-likelihood discrete choices.

To prevent particle depletion, the Effective Number of Particles (N_{eff}) is continuously monitored:

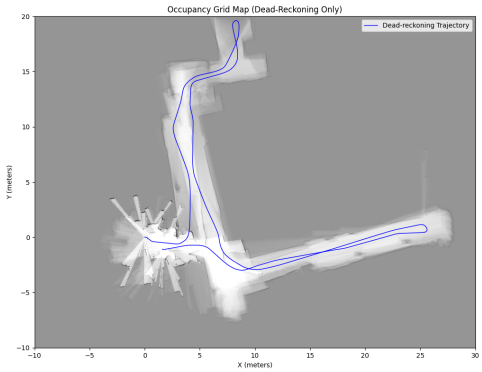
$$N_{eff} = \frac{1}{\sum_{i=1}^{N} w_i^2}$$

If N_{eff} drops below the threshold of $N/2$, the algorithm triggers a resampling step. A weighted random selection (roulette wheel selection) is performed to clone high-performing particles and eliminate those with low probabilities, after which all weights are re-initialized evenly to $1/N$.

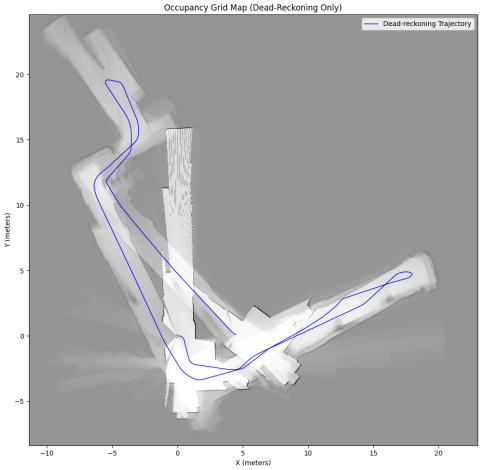
4 Results and Discussion

4.1 Resulting Maps and Trajectories

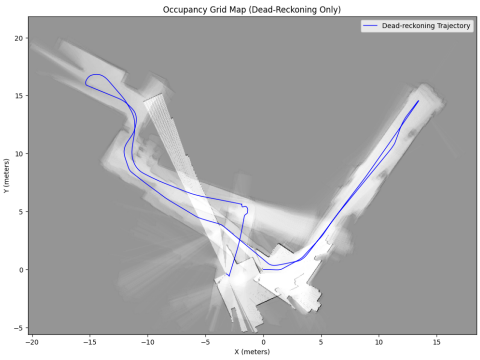
The dead-reckoning trajectory and the resulting occupancy grid maps were successfully generated using the tuned parameters across multiple datasets. Figure 2 displays the robot's motion (blue line) overlaid on the tuned maps, where dark pixels represent obstacles and white pixels represent observed free space.



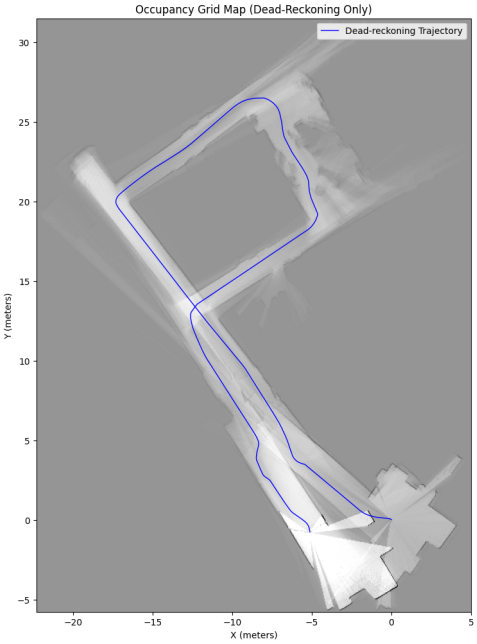
(a) Map 20



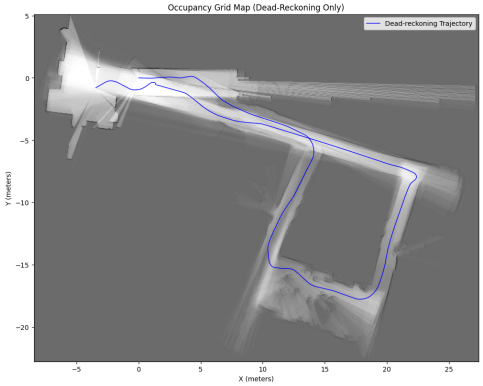
(b) Map 21



(c) Map 22



(d) Map 24



(e) Map 23

Figure 2: Occupancy grid maps generated via dead-reckoning across different runs. The blue line denotes the estimated robot trajectory.

4.2 Limitations of Dead-Reckoning

The current implementation successfully reconstructs the general geometry of the environment, capturing straight corridors and sharp corners accurately in the early stages of each run. This confirms that the coordinate transformations, ray-casting logic, and robot-width tuning are mathematically sound.

However, as observed in the outer edges of the maps (particularly in longer runs), significant structural distortions and "ghost walls" begin to appear as the trajectory progresses. This is the fundamental limitation of mapping exclusively with dead-reckoning. While tuning W_{eff} removes systematic drift, it cannot account for stochastic errors such as uneven floor friction, brief wheel slippage, or minor surface irregularities.

Because the motion model relies entirely on internal sensors (encoders) without external correction (Lidar scan matching), microscopic random errors in $\Delta\theta$ accumulate continuously. Even a 1° accumulated heading drift will cause Lidar points 10 m away to be projected with a 0.17 m translational error, causing walls to smear and diverge. When the robot maps a long continuous wall or revisits an area, this drift causes newly projected Lidar scans to physically misalign with previously mapped structures.

4.3 Comparison: Dead-Reckoning vs. Particle Filter SLAM

The Particle Filter drastically improved the global consistency and structural fidelity of the generated maps compared to pure dead-reckoning. By comparing both outputs, the mitigation of accumulated kinematic errors becomes explicitly clear across several distinct failure modes.

4.3.1 Correcting Stochastic Heading Drift (Maps 20 & 21)

In the dead-reckoning run for Map 21, the kinematic model suffered from severe, uncorrected stochastic heading drift. As the robot traveled down the top-left corridor and returned, microscopic wheel slips accumulated into a large $\Delta\theta$ error. This caused the Lidar to project the return journey at a physically impossible angle, resulting in diverging, cloned hallways.

The Particle Filter successfully neutralized this drift. By injecting proportional rotational noise during turns and tightly clamping forward translation noise on straightaways, the particles explored alternative headings. When the simulated Lidar scans were evaluated against the map, the algorithm penalized diverging paths and rewarded particles that anchored to existing walls. As seen in Figure 3b, the red trajectory corrects the blue dead-reckoning path, fusing the overlapping ghost walls into a single, crisp corridor with clean 90° corners.

4.3.2 Mitigating Erratic Wheel Slippage (Map 22)

Map 22 demonstrates the filter's ability to handle sudden, erratic odometry failures. In the dead-reckoning trajectory, the robot exhibits sharp, jagged jumps near the centre and lower-left sections of the map, likely due to brief wheel slippage or uneven floor surfaces. Left uncorrected, these jumps cause the estimated position to veer dangerously close to the walls of the long diagonal corridor.

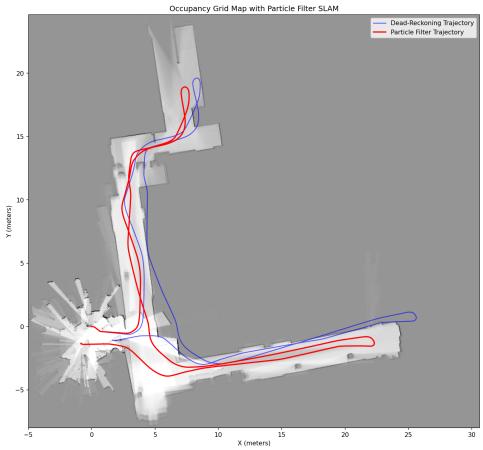
Because the Particle Filter weights particles using Lidar correlations, it inherently filters out erratic odometry jumps. Particles that blindly followed the jagged odometry

crashed into virtual walls and received near-zero weights, while particles that maintained a smooth, centred heading returned high correlation scores. The final PF trajectory (red line) smooths out the odometry spikes and accurately traces the centreline of the hallway (Figure 3c).

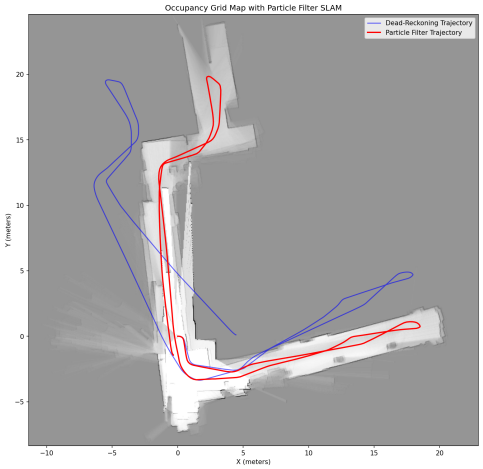
4.3.3 Catastrophic Drift Recovery and Loop Closure (Maps 23 & 24)

Maps 23 and 24 represent the most demanding stress tests: large, continuous loops where errors compound over time.

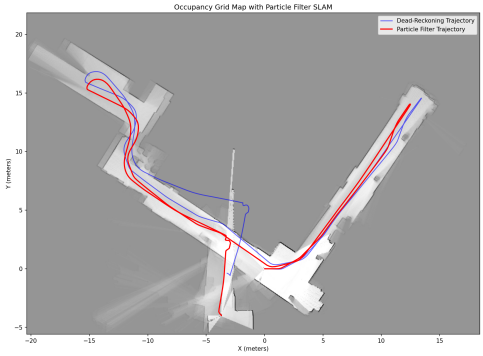
In Map 24, dead-reckoning suffered a catastrophic heading failure midway through the run. The estimated trajectory wildly diverged into empty space, failing entirely to close the rectangular loop. The Particle Filter achieved full recovery and loop closure in both datasets. Because the 15 m Lidar cap prevented the system from anchoring to distant noisy returns, particles maintained localised high-confidence tracking throughout. As the robot approached the starting area in Map 24, the particle cloud had spread sufficiently to overlap with previously mapped geometry. The update step correctly recognised this structure, assigning high weights to particles that aligned the current scan with the original walls. The red estimated trajectory actively bent away from the broken odometry line, snapping the loop shut and producing a closed rectangular floor plan (Figures 3e–3d).



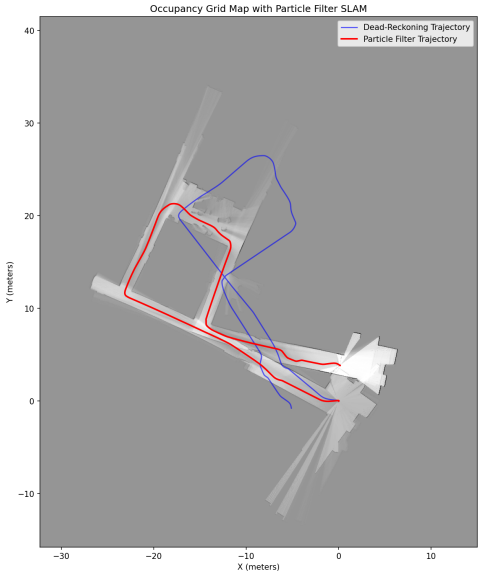
(a) PF Result: Map 20



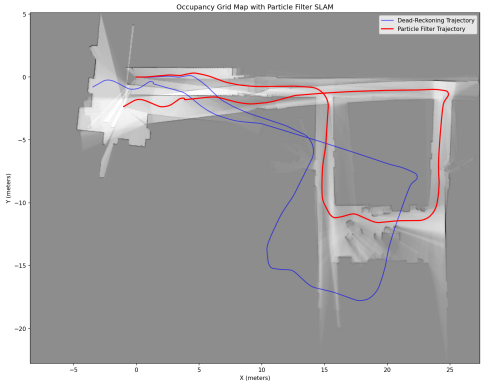
(b) PF Result: Map 21



(c) PF Result: Map 22



(d) PF Result: Map 24



(e) PF Result: Map 23

Figure 3: Occupancy grid maps generated using Particle Filter SLAM. The blue line represents the raw dead-reckoning trajectory; the red line represents the corrected SLAM trajectory.

5 Conclusion

This project successfully demonstrated the transition from a highly flawed, open-loop dead-reckoning system to a robust, closed-loop Particle Filter SLAM architecture. While manually tuning the effective wheelbase (W_{eff}) mitigated systematic wheel scrubbing errors, the dead-reckoning maps remained highly vulnerable to stochastic heading drift over long distances.

By implementing a kinematic motion model with proportional noise, enforcing a 15 m Lidar range threshold to prevent lever-arm divergence, and applying a dynamic resampling threshold ($N_{\text{eff}} < N/2$), the Particle Filter successfully tracked the robot's state across all test datasets. The final system effectively eliminated ghost walls, smoothed erratic wheel slippage, preserved sharp structural corners, and executed large-scale loop closures following catastrophic odometry failures—demonstrating the capability of probabilistic robotics in mapping unknown static environments.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[2] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.