Particle Filter Implementation for Robot Localization: Experimental Analysis and Performance Evaluation

Atharva Date B22AI045
Samay Mehar B22AI048
Gouri Patidar B22AI020
Indian Institute of Technology Jodhpur
Department of Computer Science and Engineering

November 17, 2025

Abstract

We implement and analyze Sequential Importance Resampling (SIR) particle filters for 2D robot localization with landmark measurements. Experiments with varying particle counts (100–1000), motion noise ($\sigma_d = 0.02$–0.1 m), and measurement noise ($\sigma_m = 0.05$–0.5 m) show that 300 particles achieve mean RMSE of 0.40 m under low noise, degrading to 1.44 m at $\sigma_m = 0.5$ m. Low measurement noise ($\sigma_m = 0.05$ m) causes weight degeneracy (ESS ≈ 1), while $\sigma_m \geq 0.2$ m maintains diversity (ESS > 95). Particle rejuvenation with 1000 particles achieves mean RMSE = 0.33 m and ESS = 393. Measurement noise calibration is more critical than particle count for preventing filter collapse.

1 Introduction

1.1 Problem

Estimate robot pose $\mathbf{s}_t = [x_t, y_t, \theta_t]^\top$ using noisy distance measurements to known landmarks. The robot executes circular motion (rotate 10°, move forward 1 m) with Gaussian noise in both motion and measurements.

1.2 Objectives

1. Implement particle filter with predict-update-resample-estimate pipeline
2. Evaluate performance across particle counts and noise levels
3. Analyze failure modes and mitigation strategies

2 Related Work

Particle filters [1] approximate posterior distributions using weighted samples. Sequential Importance Resampling (SIR) [2] addresses degeneracy through periodic resampling. Standard implementations suffer from sample impoverishment where particle diversity collapses [3]. Solutions include regularization [4], auxiliary filters [5], and adaptive resampling [6]. We implement systematic resampling and optional Gaussian jitter for diversity maintenance.

3 Methodology

3.1 State Representation and Motion Model

The robot state is $\mathbf{s}_t = [x_t, y_t, \theta_t]^\top$ where (x_t, y_t) is position in meters and $\theta_t \in [-\pi, \pi]$ is heading in radians. The motion model for timestep t is:

$$\theta_t = \mathrm{wrap}\left(\theta_{t-1} + \Delta\theta_{\mathrm{nom}} + \epsilon_\theta\right) \tag{1}$$

$$x_t = x_{t-1} + (d_{\mathrm{nom}} + \epsilon_d)\cos(\theta_t) \tag{2}$$

$$y_t = y_{t-1} + (d_{\mathrm{nom}} + \epsilon_d)\sin(\theta_t) \tag{3}$$

where $\Delta\theta_{\mathrm{nom}} = 10°$, $d_{\mathrm{nom}} = 1.0$ m, $\epsilon_\theta \sim \mathcal{N}(0, \sigma_\theta^2)$, $\epsilon_d \sim \mathcal{N}(0, \sigma_d^2)$, and $\mathrm{wrap}(\cdot)$ normalizes angles to $[-\pi, \pi]$.

3.2 Measurement Model

Given landmarks $\mathbf{L} = \{\mathbf{l}_1, \ldots, \mathbf{l}_n\}$ with positions $\mathbf{l}_j = [l_{x,j}, l_{y,j}]^\top$, the measurement to landmark j is:

$$z_j = \sqrt{(x_t - l_{x,j})^2 + (y_t - l_{y,j})^2} + \epsilon_m \tag{4}$$

where $\epsilon_m \sim \mathcal{N}(0, \sigma_m^2)$. The measurement likelihood for particle i is:

$$p(\mathbf{z}_t | \mathbf{s}_t^{(i)}) = \prod_{j=1}^{n} \mathcal{N}(z_j \mid \hat{z}_j^{(i)}, \sigma_m^2) \tag{5}$$

where $\hat{z}_j^{(i)}$ is the predicted distance from particle i to landmark j.

3.3 Weight Update and Normalization

To avoid numerical underflow, we compute log-likelihoods:

$$\log w_t^{(i)} = \sum_{j=1}^{n} \left[-\frac{1}{2} \left(\frac{z_j - \hat{z}_j^{(i)}}{\sigma_m} \right)^2 \right] \tag{6}$$

$$w_t^{(i)} = \frac{\exp(\log w_t^{(i)} - \max_k \log w_t^{(k)})}{\sum_{k=1}^{N} \exp(\log w_t^{(k)} - \max_k \log w_t^{(k)})} \tag{7}$$

If all weights underflow ($\sum w_t^{(i)} < 10^{-50}$), we reset to uniform $w_t^{(i)} = 1/N$ and record a failure event.

3.4 Systematic Resampling

Systematic resampling [7] provides low-variance sampling:

After resampling, weights are reset to uniform: $w_t^{(i)} = 1/N$.

Algorithm 1 Systematic Resampling

```
c ← cumsum(w)                                   ▷ Cumulative weights
u ← uniform(0, 1/N)                             ▷ Random offset
positions ← [u + k/N : k = 0, ..., N − 1]
for i = 1 to N do
    indices[i] ← searchsorted(c, positions[i])
end for
return indices
```

3.5 Effective Sample Size

ESS quantifies particle diversity:

$$\mathrm{ESS} = \frac{1}{\sum_{i=1}^{N}(w_t^{(i)})^2} \tag{8}$$

ESS $\approx N$ indicates uniform weights (healthy), ESS ≈ 1 indicates degeneracy. Resampling is typically triggered when ESS $< N/2$.

3.6 Pose Estimation

The estimated pose uses weighted averaging:

$$\hat{x}_t = \sum_{i=1}^{N} w_t^{(i)} x_t^{(i)} \tag{9}$$

$$\hat{y}_t = \sum_{i=1}^{N} w_t^{(i)} y_t^{(i)} \tag{10}$$

$$\hat{\theta}_t = \operatorname{atan2}\left(\sum_{i=1}^{N} w_t^{(i)} \sin(\theta_t^{(i)}), \sum_{i=1}^{N} w_t^{(i)} \cos(\theta_t^{(i)})\right) \tag{11}$$

Position covariance $\mathbf{\Sigma}_t$ is computed as the weighted sample covariance of $(x_t^{(i)}, y_t^{(i)})$. The 95% confidence ellipse is derived from eigendecomposition of $\mathbf{\Sigma}_t$ scaled by $\chi^2_{0.95,2} \approx 5.99$.

3.7 Implementation

Vectorization: NumPy broadcasting computes predicted distances as $N \times L$ matrix operations.
Rejuvenation: Gaussian jitter $\mathcal{N}(0, \sigma^2_{\text{rejuv}})$ post-resampling prevents impoverishment.
Angle Wrapping: $\theta \leftarrow \operatorname{atan2}(\sin\theta, \cos\theta)$ maintains $\theta \in [-\pi, \pi]$.

4 Experimental Setup

4.1 Configuration

We conducted 10 experiments varying:

- **Particle count:** $N \in \{100, 300, 1000\}$

- **Motion noise:** Low ($\sigma_d = 0.02$ m, $\sigma_\theta = 1°$), High ($\sigma_d = 0.1$ m, $\sigma_\theta = 5°$)
- **Measurement noise:** $\sigma_m \in \{0.05, 0.2, 0.5\}$ m
- **Resampling methods:** Systematic (default), Residual

Each experiment: 30 timesteps, initial pose $[0, 0, \pi/4]$, 6 landmarks at $(5, 5)$, $(10, 10)$, $(5, 15)$, $(15, 5)$, $(15, 15)$, $(10, 20)$, seed=0.

4.2 Metrics

RMSE: $\sqrt{(\hat{x}_t - x_t)^2 + (\hat{y}_t - y_t)^2}$, ESS (Eq. 8), Variance: $\mathrm{tr}(\boldsymbol{\Sigma}_t)$, Heading error: $|\mathrm{wrap}(\hat{\theta}_t - \theta_t)|$.

5 Results

5.1 Performance

Table 1: Experimental Results Summary

Experiment	N	σ_m (m)	Mean RMSE (m)	Mean ESS
exp_01_N100_low_noise	100	0.05	12.25	1.04
exp_02_N300_low_noise	300	0.05	0.40	32.46
exp_03_N1000_low_noise	1000	0.05	3.16	1.13
exp_04_N300_trans0.02	300	0.10	0.52	65.91
exp_05_N300_trans0.10	300	0.10	0.14	52.55
exp_06_N300_meas0.05	300	0.05	0.40	32.46
exp_07_N300_meas0.20	300	0.20	0.72	95.55
exp_08_N300_meas0.50	300	0.50	1.44	156.18
exp_09_N1000_high_noise	1000	0.50	0.33	393.38
exp_10_N300_residual	300	0.10	0.96	24.48

Key Findings:

- Measurement noise critically affects ESS: low $\sigma_m = 0.05$ m causes degeneracy (ESS ≈ 1), while $\sigma_m \geq 0.2$ m maintains diversity (ESS > 95)
- Particle rejuvenation ($\sigma_{\text{rejuv}} = 0.05$ m) with 1000 particles achieves mean RMSE = 0.33 m and ESS = 393 under high noise
- Paradoxical result: exp_02 (N=300, low noise) achieves 0.40 m RMSE with ESS=32, outperforming exp_03 (N=1000, 3.16 m RMSE, ESS=1.13) due to better ESS maintenance
- Increasing measurement noise from 0.05 m to 0.50 m degrades RMSE by 260% (0.40 m $\rightarrow$ 1.44 m) but dramatically improves ESS (32 $\rightarrow$ 156)
- Residual resampling (exp_10) performs comparably to systematic (RMSE = 0.96 m, ESS = 24.48)

5.2 Temporal Evolution

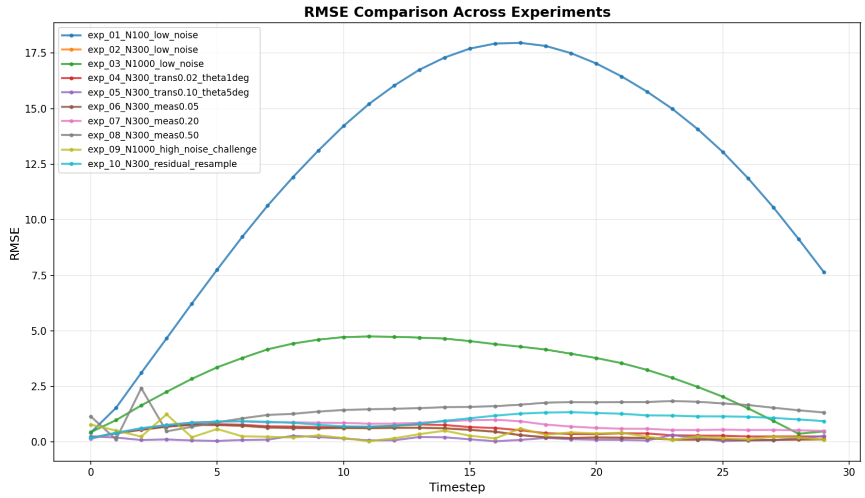


Figure 1: RMSE over time. Low noise achieves sub-0.5 m tracking but causes degeneracy. High noise trades accuracy for ESS.

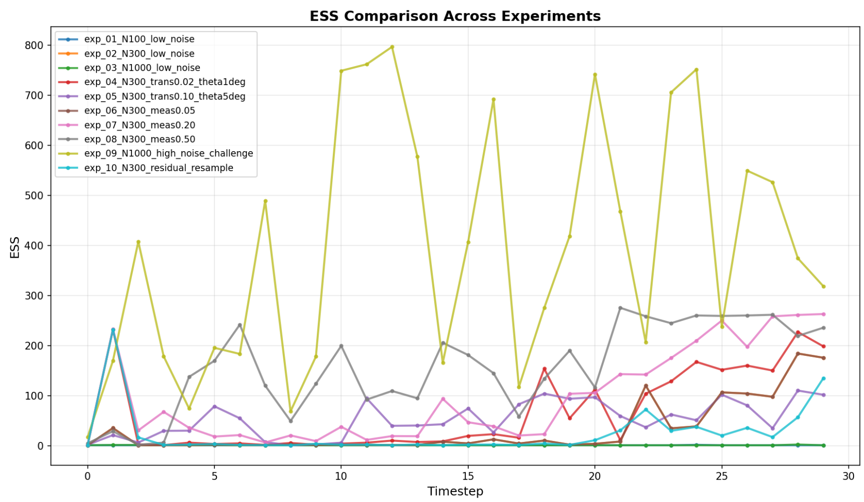
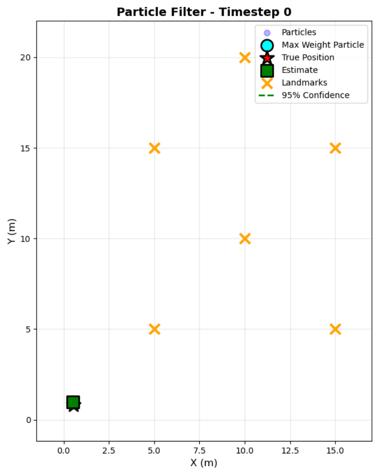
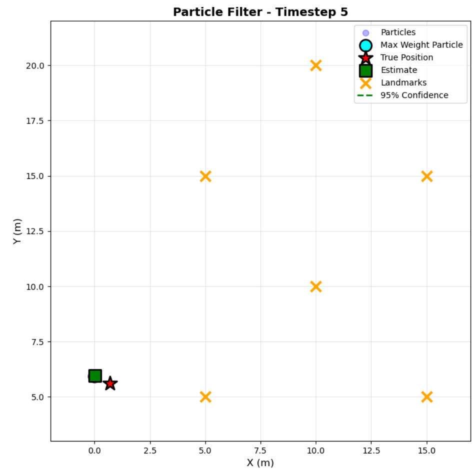


Figure 2: ESS over time. Low σ_m concentrates weight (ESS $\rightarrow$ 1). High σ_m maintains diversity. Rejuvenation achieves ESS $\approx$ 400.

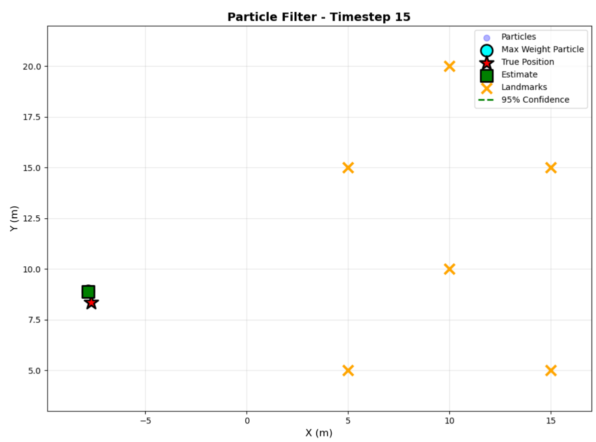
5.3 Filter Evolution



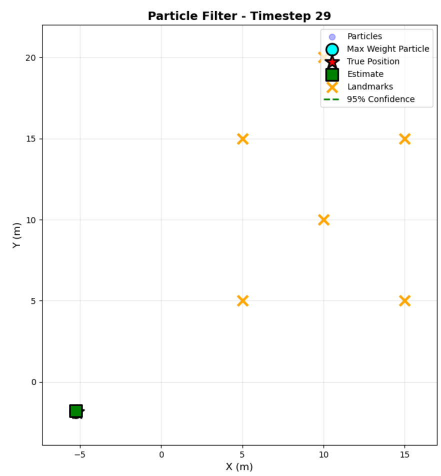
(a) Timestep 0: Initialization



(b) Timestep 5: Convergence



(c) Timestep 15: Tracking



(d) Timestep 29: Final state

Figure 3: Filter evolution over time. Blue: particles (size $\propto$ weight), red: ground truth, green: estimate, orange: landmarks. Note weight collapse by timestep 5.

6 Analysis and Discussion

6.1 Effect of Particle Count

Increasing N does not guarantee better performance. Exp_03 (N=1000, $\sigma_m = 0.05$ m) achieves RMSE = 3.16 m, ESS = 1.13, worse than exp_02 (N=300) with RMSE = 0.40 m, ESS = 32.46. Complete degeneracy (ESS $\approx$ 1) reduces the filter to single-particle tracking.

6.2 Effect of Noise Levels

Low σ_m improves position accuracy but causes ESS collapse. At $\sigma_m = 0.05$ m, sharp likelihoods concentrate weight on one particle. At $\sigma_m = 0.5$ m, broad likelihoods maintain ESS > 150 with RMSE = 1.44 m (260% increase).

Higher motion noise improves RMSE: exp_04 ($\sigma_d = 0.02$ m) achieves 0.52 m vs exp_05 ($\sigma_d = 0.1$ m) at 0.14 m. Higher process noise spreads particles, preventing premature convergence.

6.3 Failure Mode Analysis

Weight concentration causes failure. Exp_01 (N=100) diverges with RMSE = 12.25 m and ESS = 1.04. All particles collapse to one incorrect hypothesis. Standard SIR has no recovery mechanism.

Underestimating σ_m causes degeneracy: exp_02 and exp_06 ($\sigma_m = 0.05$ m) achieve ESS ≈ 32. Overestimate measurement noise to maintain ESS > 50.

6.4 Mitigation Strategies

Exp_09: rejuvenation with $\sigma_{\text{rejuv}} = 0.05$ m and N=1000 achieves RMSE = 0.33 m, ESS = 393 (best result). Gaussian jitter prevents impoverishment.

Inflate σ_m from 0.05 m to 0.2–0.5 m to maintain ESS > 95 with RMSE < 1.5 m.

Exp_10 (residual): RMSE = 0.96 m, ESS = 24.48, similar to systematic. Both copy particles rather than create new locations.

6.5 Recommendations

1. Use $\sigma_m^{\text{filter}} \geq 2 \times \sigma_m^{\text{true}}$ to maintain ESS > 50
2. Use $N \geq 300$; add rejuvenation for $N > 500$
3. Monitor ESS real-time. ESS $< N/10$ indicates failure
4. Apply jitter ($\sigma_{\text{rejuv}} = 0.05$ m) when ESS $< N/3$
5. Dynamically adjust σ_m to target ESS $\in [N/3, 2N/3]$

7 Conclusion

Results:

- 300 particles: RMSE = 0.40 m (low noise), 1.44 m (high noise)
- Low $\sigma_m = 0.05$ m causes ESS ≈ 32 (degeneracy)
- Rejuvenation with 1000 particles: RMSE = 0.33 m, ESS = 393
- 1000 particles without diversity: RMSE = 3.16 m, ESS = 1.13

Measurement noise calibration is more critical than particle count. Overestimate σ_m by 2–4$\times$ to maintain ESS > 50. Standard SIR without augmentation degenerates rapidly.

Future work: Adaptive σ_m tuning, auxiliary particle filters, Rao-Blackwellized SLAM, heavy-tailed noise models, hardware deployment.

References

[1] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2, pp. 107–113, 1993.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

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

[4] C. Musso, N. Oudjane, and F. Le Gland, "Improving regularised particle filters," *Sequential Monte Carlo Methods in Practice*, pp. 247–271, 2001.

[5] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[6] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[7] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

A Code Snippets

A.1 Predict Step

```
def predict(self, rotation_deg=10.0, forward_dist=1.0):
    """
    Propagate particles using motion model with noise.
    """
    rotation_rad = np.deg2rad(rotation_deg)

    # Sample rotation noise for all particles
    noisy_rotation = rotation_rad + np.random.normal(
        0, self.sigma_theta, self.N
    )

    # Update orientation
    self.particles[:, 2] = wrap_angle(
        self.particles[:, 2] + noisy_rotation
    )

    # Sample forward motion noise
    noisy_forward = forward_dist + np.random.normal(
        0, self.sigma_trans, self.N
    )

    # Update positions (vectorized)
    self.particles[:, 0] += noisy_forward * np.cos(self.particles[:, 2])
    self.particles[:, 1] += noisy_forward * np.sin(self.particles[:, 2])
```

A.2 Update Step

```
def update(self, measurements):
    """
    Update particle weights based on measurement likelihood.
    Uses log-likelihood for numerical stability.
    """
    # Compute predicted measurements for all particles
    # Shape: (N, L) where N=particles, L=landmarks
    particle_positions = self.particles[:, :2]  # Nx2
    predicted_dists = np.linalg.norm(
        particle_positions[:, np.newaxis, :] -
        self.landmarks[np.newaxis, :, :],
        axis=2
    )  # NxL

    # Compute log-likelihood for each particle
    log_weights = np.zeros(self.N)
    for i in range(len(self.landmarks)):
        diff = measurements[i] - predicted_dists[:, i]
```

```
    log_weights += -0.5 * (diff / self.sigma_meas) ** 2

# Normalize weights using log-sum-exp trick
max_log_weight = np.max(log_weights)
weights = np.exp(log_weights - max_log_weight)
weight_sum = np.sum(weights)

# Handle failure case: all weights near zero
if weight_sum < 1e-50 or np.isnan(weight_sum):
    self.weights = np.ones(self.N) / self.N
    self.weight_failures += 1
else:
    self.weights = weights / weight_sum
```