Advanced Mathematical Analysis
of Six SLAM Algorithms

Ram Anurag
ramanurag@ece.du.ac.in

Overview

This document summarizes the mathematical intuition and optimization principles behind six localization algorithms forming the foundation of the *Advanced Multi-Algorithm SLAM* pipeline. Each section introduces the conceptual approach, derives simplified core equations, and highlights how optimization improves accuracy, stability, and convergence.

1 Dead-Reckoning

Concept: A fundamental motion-based localization approach that integrates wheel encoder and gyro data to estimate pose incrementally.

Kinematic Model: Robot pose at time t is represented as $\mathbf{x}_t = [x_t,\, y_t,\, \theta_t]^T$. The motion update is derived from differential drive kinematics:

$$\dot{x}_t = v_t \cos(\theta_t), \quad \dot{y}_t = v_t \sin(\theta_t), \quad \dot{\theta}_t = \omega_t$$

Discretizing over a small timestep Δt:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_t + v_t \cos(\theta_t)\Delta t \\ y_t + v_t \sin(\theta_t)\Delta t \\ \theta_t + \omega_t \Delta t \end{bmatrix}$$

Incremental Pose Update:

$$\Delta \mathbf{x}_t = \begin{bmatrix} v_t \cos\theta_t \\ v_t \sin\theta_t \\ \omega_t \end{bmatrix} \Delta t$$

Hence,

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t + \Delta \mathbf{x}_t$$

Error Propagation: Assuming process noise $\mathbf{w}_t \sim \mathcal{N}(0, Q_t)$,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t$$

Drift increases with time as:

$$\sigma_{error}(t) \propto \sqrt{t} \quad \text{(random walk)}, \quad \text{bias drift} \propto t$$

Optimization Ideology

Errors grow with time: $\mathcal{O}(\sqrt{t})$ for noise, $\mathcal{O}(t)$ for bias. Optimization achieved through gyro calibration, drift correction, and fusing GPS or LiDAR feedback.

2 Extended Kalman Filter (EKF)

Concept: Linearizes nonlinear motion and measurement models for real-time probabilistic estimation.

Prediction Step:

Nonlinear Motion Model:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, Q_t)$$

Linearizing about the current estimate:

$$F_t = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}}$$

Predicted mean and covariance:

$$\hat{\mathbf{x}}_{t|t-1} = f(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}), \quad P_{t|t-1} = F_t P_{t-1} F_t^\top + Q_t$$

Interpretation: The Jacobian F_t maps previous uncertainty forward through system dynamics.

Update Step:

Measurement Model:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(0, R_t)$$

Linearization around the predicted state:

$$H_t = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t|t-1}}$$

Compute innovation (measurement residual):

$$\mathbf{y}_t = \mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1})$$

Kalman gain and corrected estimates:

$$K_t = P_{t|t-1} H_t^\top (H_t P_{t|t-1} H_t^\top + R_t)^{-1}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + K_t \mathbf{y}_t, \quad P_{t|t} = (I - K_t H_t) P_{t|t-1}$$

Consistency: Innovation covariance:

$$S_t = H_t P_{t|t-1} H_t^\top + R_t$$

Optimization Ideology

Optimization via adaptive noise tuning and iterated EKF enhances stability under nonlinear motions, balancing computational cost with accuracy.

3 Unscented Kalman Filter (UKF)

Concept: Replaces Jacobian linearization with deterministic sigma points for higher non-linear precision.

Unscented Transform & Sigma-Point UKF (compact derivation)

Parameters: state dimension n, scaling α, spread κ, prior weight β. Define $\lambda = \alpha^2(n+\kappa) - n$.

Sigma-point generation:

$$\chi_0 = \hat{\mathbf{x}}, \qquad \chi_i = \hat{\mathbf{x}} + \left[\sqrt{(n+\lambda)P}\right]_i, \quad \chi_{i+n} = \hat{\mathbf{x}} - \left[\sqrt{(n+\lambda)P}\right]_i,$$

for $i = 1, \ldots, n$, where $\sqrt{\cdot}$ denotes the matrix square root (e.g. Cholesky).

Weights:

$$W_0^{(m)} = \frac{\lambda}{n+\lambda}, \quad W_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta)$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(n+\lambda)}, \qquad i = 1, \ldots, 2n.$$

Predicted mean and covariance:

$$\hat{\mathbf{x}}^- = \sum_{i=0}^{2n} W_i^{(m)} \chi_i^-$$

$$P^- = \sum_{i=0}^{2n} W_i^{(c)} (\chi_i^- - \hat{\mathbf{x}}^-)(\chi_i^- - \hat{\mathbf{x}}^-)^\top + Q$$

(ensure angle components are wrapped when computing differences)

Measurement prediction: propagate sigma points through $h(\cdot)$:

$$\zeta_i = h(\chi_i^-), \qquad \hat{\mathbf{z}} = \sum_{i=0}^{2n} W_i^{(m)} \zeta_i$$

$$S = \sum_{i=0}^{2n} W_i^{(c)} (\zeta_i - \hat{\mathbf{z}})(\zeta_i - \hat{\mathbf{z}})^\top + R$$

State update:

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^- + K(\mathbf{z} - \hat{\mathbf{z}}), \qquad P = P^- - KSK^\top$$

(again, wrap angular residuals appropriately)

Notes: choose (α, κ, β) for a balance of spread and higher-order moment matching (typical defaults: $\alpha \in [10^{-3}, 1]$, $\kappa = 0$, $\beta = 2$ for Gaussian priors).

Optimization Ideology

Performance governed by parameters α, κ, β. Provides smoother transitions — ideal for drones and agile robotic systems.

4 Quantum Particle Swarm Optimization (QPSO)

Concept: A metaheuristic, derivative-free optimization algorithm for refining poses in global search spaces.

$$\mathbf{p}_i^{new} = \mathbf{p}_{att} + \beta|\mathbf{mbest} - \mathbf{p}_i| \odot \ln(1/\mathbf{u}), \quad J = \sum \frac{(z_{measured} - z_{expected})^2}{\sigma^2}$$

Optimization Ideology

Quantum potential field guides convergence. β controls exploration–exploitation balance, making QPSO robust to non-convex landscapes and sensor noise.

5 Neural Network Residual Learning (LSTM)

Concept: A hybrid approach where an LSTM learns to predict residual errors in traditional state estimation models.

$$\hat{\mathbf{x}}_{t+1} = f(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \mathrm{NN}([\hat{\mathbf{x}}_t; \mathbf{u}_t])$$

Optimization Ideology

Trained on motion–sensor data with MSE loss. Improves adaptability and robustness over time — ideal for long-term autonomous deployment.

6 GraphSLAM

Concept: Batch optimization builds a graph of poses and landmarks, minimizing all motion and measurement residuals jointly.

GraphSLAM — Batch Pose Graph Optimization (compact derivation)

Objective (restated):

$$\min_{\mathbf{x}} \; J(\mathbf{x}) \;=\; \sum_{t} \|\mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t)\|^2_{Q_t} \;+\; \sum_{t,m} \|\mathbf{z}_{t,m} - h(\mathbf{x}_t, \ell_m)\|^2_{R_{t,m}}$$

where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^\top$ is the stacked trajectory and ℓ_m are known/estimated landmarks.

Error terms (pose–pose and pose–landmark):

$$e_t^{\text{odom}}(\mathbf{x}_t, \mathbf{x}_{t+1}) = \mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t),$$

$$e_{t,m}^{\text{meas}}(\mathbf{x}_t) = \mathbf{z}_{t,m} - h(\mathbf{x}_t, \ell_m).$$

Linearization (Gauss–Newton step): linearize each error about current estimate $\mathbf{x}^{(k)}$:

$$e \approx e(\mathbf{x}^{(k)}) + J_e(\mathbf{x}^{(k)})\, \delta\mathbf{x}$$

Stack all linearized residuals to form normal equations:

$$H\, \delta\mathbf{x} = -b,$$

Structure and sparse solve: H is block-sparse (3×3 blocks for 2D poses). Solve efficiently using sparse Cholesky / QR:

$$\delta\mathbf{x} = -H^{-1}b \quad \text{(computed via sparse factorization)}$$

Optimization Ideology:

- Use **robust kernels** (e.g., Huber, Tukey) on measurement residuals to reduce influence of outliers before forming H.
- Initialize with odometry/EKF trajectory for faster convergence (good initial guess reduces nonlinearity issues).
- For real-time systems, run incremental solvers (iSAM2) that incrementally update factorization on each new constraint instead of full batch re-solve.

Optimization Ideology

Employs Gauss–Newton or Levenberg–Marquardt for optimization. Achieves global consistency through loop closures, though with higher computational load.

Summary Insight

Algorithm	Strength	Limitation	Ideal Use-Case
Dead-Reckoning	Simple, fast	Drift accumulates	Low-cost navigation
EKF	Real-time optimal	Linearization errors	Ground / warehouse robots
UKF	Nonlinear precision	More computation	Agile drones, fast turns
QPSO	Global search	Stochastic, slow	Rough terrain mapping
Neural-Net	Learns residuals	Needs data	Adaptive long-term systems
GraphSLAM	Globally optimal	High computation	Offline mapping / surveys

Takeaway

From incremental estimation to full graph optimization, each method balances accuracy, speed, and adaptability uniquely. Their combined implementation defines the architecture of modern autonomous SLAM systems.