FastSLAM

Solving the Chicken-and-Egg Problem

The Core Challenge:

"To map the world accurately, the robot needs to know where it is."

"But to figure out where it is, the robot needs a good map of the world."

The Solution: SLAM

"The ability to simultaneously localize a robot and accurately map its environment is considered by many to be a key prerequisite of truly autonomous robots."

FastSLAM Approach:

- Uses particle filters to track multiple location hypotheses
- Breaks the problem into manageable sub-problems
- Efficiently solves the chicken-and-egg paradox

FastSLAM: The Conditional Independence Property

$$p(s_t, \theta \mid z_t, u_t, n_t) = p(s_t \mid z_t, u_t, n_t) \prod k p(\theta k \mid s_t, z_t, u_t, n_t)$$

Where: s_t = robot pose, θ = landmarks, z_t = measurements, u_t = controls, n_t = data associations

The Factorization

1. Localization Problem:

$p(s_t \mid z_t, u_t, n_t)$

Sampling of robot poses using particle filter

2. Mapping Problem:

$\prod k \ p(\theta k \mid s_t, z_t, u_t, n_t)$

Estimating landmarks given a known path

Each particle maintains separate EKFs for each landmark

Why It Matters

- Transforms high-dimensional SLAM into manageable subproblems
- Each landmark becomes conditionally independent
- Enables efficient parallel computation
- Uses small 2×2 Gaussian distributions per landmark

Key Insight:

Once the robot's pose is known, all landmark estimations become independent problems that can be solved separately

One EKF Per Landmark

In FastSLAM, each particle maintains its own set of landmark estimates:

$$S_t = \{s_{t\,[m]}, \mu_{[m]}1, \Sigma_{[m]}1, \ldots, \mu_{[m]}K, \Sigma_{[m]}K\}m$$

- Each particle carries **full robot path** plus **K landmark estimates**
- Each landmark estimate represented by a **Gaussian distribution**:
  - μ: Mean (x,y position)
  - Σ: Covariance (uncertainty)
- Landmark estimates are **conditioned on the particle's path**
- For M particles and K landmarks: **M×K independent EKFs**

Key Insight:
Given a known robot path, landmark positions become independent of each other!

Why This Matters

Traditional EKF SLAM

Single joint Gaussian

- State: robot + all landmarks
- Full covariance matrix
- $O(K^2)$ update complexity
- Update all landmarks

FastSLAM

M particles × K small EKFs

- Particle: robot path
- Small 2×2 covariances
- O(M log K) complexity
- Update only observed landmarks

Using separate EKFs per landmark allows FastSLAM to:

- Handle **non-linear motion models** (via particle filter)
- Maintain **multiple data association hypotheses**
- Scale to **large numbers of landmarks**

Recursive Bayes Filter

Definition $Bel(x_t) = P(x_t | u_1, z_1, \ldots, u_t, z_t)$

Bayes $= \eta\, P(z_t | x_t, u_1, z_1, \ldots, u_t)\, P(x_t | u_1, z_1, \ldots, u_t)$

Markov $= \eta\, P(z_t | x_t)\, P(x_t | u_1, z_1, \ldots, u_t)$

Total prob. $= \eta\, P(z_t | x_t) \int P(x_t | u_1, z_1, \ldots, u_t, x_{t-1})\, P(x_{t-1} | u_1, z_1, \ldots, u_t)\, dx_{t-1}$

Markov $= \eta\, P(z_t | x_t) \int P(x_t | u_t, x_{t-1})\, P(x_{t-1} | u_1, z_1, \ldots, u_t)\, dx_{t-1}$

Markov $= \eta\, P(z_t | x_t) \int P(x_t | u_t, x_{t-1})\, P(x_{t-1} | u_1, z_1, \ldots, z_{t-1})\, dx_{t-1}$

Recursive $= \eta\, \underline{P(z_t | x_t)} \int \underline{P(x_t | u_t, x_{t-1})}\, Bel(x_{t-1})\, dx_{t-1}$

Correction Step / Observation Model — Prediction Step / Motion Model

The Bayes Filter is a framework for recursive state estimation:

- Linear vs non-linear models
- Gaussian vs non gaussian distributions
- Parametric vs non parametric filters

Kalman Filter Algorithm

Prediction:

1. $\bar{\mu}_k = A_k\mu_{k-1} + B_ku_k$
2. $\bar{\Sigma}_k = A_k\Sigma_{k-1}A_k^T + R_k$

Correction:

3. $K_k = \bar{\Sigma}_kC_k^T(C_k\bar{\Sigma}_kC_k^T + Q_k)^{-1}$
4. $\mu_k = \bar{\mu}_k + K_k(z_k - C_k\bar{\mu}_k)$
5. $\Sigma_k = (I - K_kC_k)\bar{\Sigma}_k$

6. Return μ_k, Σ_k

Parameters:

μ_{k-1}, Σ_{k-1}: Previous state mean and covariance

u_k: Control input

z_k: Measurement

A_k: State transition model

B_k: Control input model

C_k: Observation model

R_k: Process noise covariance

Q_k: Measurement noise covariance

K_k: Kalman gain

Key Insight:

The Kalman filter is a recursive estimator that provides optimal estimates of the state of a linear dynamic system from noisy measurements.

It operates in two steps: prediction and correction, following the Bayes filter framework.

Extended Kalman Filter Algorithm

Prediction:

1. $\bar{\mu}_k = g(\mu_{k-1}, u_k)$
2. $\bar{\Sigma}_k = A_k \Sigma_{k-1} A_k^T + R_k$

Where $A_k = \partial g/\partial x|_{x=\mu_{k-1}}$ (Jacobian of g)

Correction:

3. $K_k = \bar{\Sigma}_k H_k^T (H_k \bar{\Sigma}_k H_k^T + Q_k)^{-1}$

Where $H_k = \partial h/\partial x|_{x=\bar{\mu}_k}$ (Jacobian of h)

4. $\mu_k = \bar{\mu}_k + K_k(z_k - h(\bar{\mu}_k))$
5. $\Sigma_k = (I - K_k H_k)\bar{\Sigma}_k$

6. Return μ_k, Σ_k

Parameters:

μ_{k-1}, Σ_{k-1}: Previous state mean and covariance

u_k: Control input

z_k: Measurement

g: Non-linear state transition function

h: Non-linear measurement function

A_k: Jacobian of state transition function

H_k: Jacobian of measurement function

R_k: Process noise covariance

Q_k: Measurement noise covariance

K_k: Kalman gain

Key Insight:

The Extended Kalman Filter handles non-linear systems by linearizing around the current state estimate.

Unlike standard KF, it uses:

- Non-linear functions g and h
- Linearization via Jacobian matrices
- First-order Taylor approximation

The EKF works well when:

- Non-linearities are mild
- Updates are frequent
- Uncertainties remain small

Mapping in FastSLAM with EKF

Function *h* given the landmark position m_{ct} and the robot pose x_{kt} is linearized using a Taylor expansion:

$$h(m_{ct}, x_{kt}) \approx z_{kt} + H_{kt}(m_{ct} - \mu_k ct,t-1) \tag{1}$$

Here, z_{kt} is the predicted measurement, $\mu_k ct,t-1$ is the current estimate of the landmark position and H_{kt} is the Jacobian of *h*.

Using this linear approximation, the EKF updates the landmark's position and uncertainty. The Kalman gain K_{kt} is calculated to determine the influence of the new measurement:

Mapping in FastSLAM with EKF (cont.)

EKF Correction Step

$$K_{kt} = \Sigma_k ct,t\text{-}1 (H_{kt})^T (H_{kt} \Sigma_k ct,t\text{-}1 (H_{kt})^T + Q_t)^{-1} \quad (2)$$

Here, $\Sigma_k ct,t\text{-}1$ is the covariance of the landmark estimate from the previous step, and Q_t is the measurement noise covariance.

The new mean $\mu_k ct,t$ is updated with the new measurement z_t:

$$\mu_k ct,t = \mu_k ct,t\text{-}1 + K_{kt}(z_t - z_{kt}) \quad (3)$$

where z_t is the actual measurement observed by the robot, and z_{kt} is the predicted measurement.

Finally, the covariance $\Sigma_k ct,t$ of the landmark position is updated to reflect the new estimate:

$$\Sigma_k ct,t = (I - K_{kt} H_{kt}) \Sigma_k ct,t\text{-}1 \quad (4)$$

where I is the identity matrix.

Particle Importance Weights in FastSLAM

Weight Calculation

Starting with:

$$w_{t\,[m]} \propto \frac{p(s_{t\,[m]} \mid z_t, u_t, n_t)}{p(s_{t\,[m]} \mid z_{t-1}, u_t, n_{t-1})}$$

Through Bayes and Markov:

$$\propto p(z_t \mid \theta, s_{t\,[m]}, n_t)$$

Using EKF approximation:

$$\approx \int p(z_t \mid \theta_{n\,[m]}, s_{t\,[m]}, n_t)\, p(\theta_{n\,[m]})\, d\theta_n$$

Key Insight:

Particle weights are proportional to how well they explain the current observation relative to their predicted position

Weight Interpretation

- Weights determine which particles are likely to survive resampling
- Particles that better predict observations receive higher weights
- Integration can be computed in closed form for linear Gaussian models
- The EKF linearization allows efficient weight calculation

Practical Implementation:

For each particle:

1. Predict landmark position from particle path
2. Compare observed landmark position with prediction
3. Weight is higher when prediction matches observation
4. Calculate using Gaussian probability density function

Monte Carlo Localization (MCL)

S^{t-1}

Previous sample set

Prediction
+
Correction

$\bar{S}^{t}$

Temporary sample set

Resampling

S^{t}

Final sample set

Algorithm MCL(S^{t-1}, u^{t}, z^{t},):

  $S^{t} = \bar{S}^{t} = \emptyset$

  for m = 1 to M do

    $s^{t}[m]$ = sample_motion_model(u^{t}, $s^{t-1}[m]$)

    $w^{t}[m]$ = measurement_model(z^{t}, $s^{t}[m]$)

    $\bar{S}^{t} = \bar{S}^{t} + \langle s^{t}[m], w^{t}[m] \rangle$

  endfor

  for m = 1 to M do

    draw i with probability $\propto w^{t}[i]$

    add $s^{t}[i]$ to S^{t}

  endfor

  return S^{t}

$$s_t^{\{[m]\}} \sim p\left(s_t \,\middle|\, u_t, s_{\{t-1\}}^{\{[m]\}}\right)$$

- Each particle represents a possible robot state (position and orientation)
- Particles are propagated through the motion model, then weighted by measurement likelihood
- Resampling focuses computational resources on the most likely states

FastSLAM Algorithm

Initialize Particles

Create M particles with initial pose and empty map

↓

For each timestep:

1. Particle Motion Update

- Sample new pose for each particle using motion model
- $s_t^{[m]} \sim p(s_t \mid u_t, s_{t-1}^{[m]})$

↓

2. Landmark Observation

- Process sensor measurement z_t
- Identify landmark n_t (data association)

↓

3. Update Landmark Estimates with EKF

- For each particle $m = 1$ to M:
  - Compute Kalman gain K_t^k
  - Update mean $\mu_{ct,t}^k$
  - Update covariance $\Sigma_{ct,t}^k$
  - Update tree structure ($O(\log K)$ complexity)

↓

4. Calculate Importance Weights

- $w_t^{[m]}$ based on measurement likelihood

↓

5. Resampling

- Draw M new particles with probability $\propto w_t^{[m]}$
- Focus computational resources on most likely states

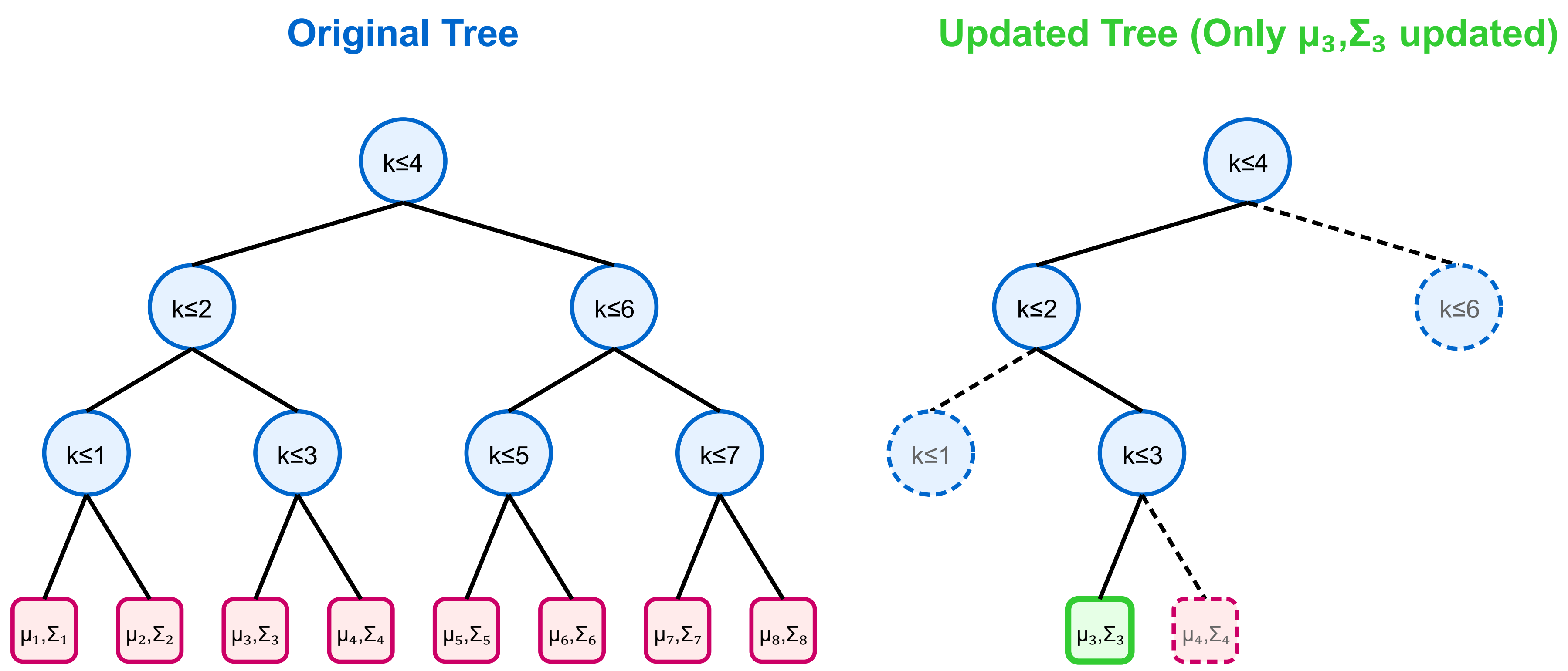
↓

Output Estimate

Most likely robot trajectory and map from highest-weight particle

Binary Tree Structure

O(M log K) Efficient Implementation



Key Benefits:

- Only path to updated landmark is modified - O(log K) complexity
- Unmodified subtrees are reused through pointers

Data Association in FastSLAM

The Correspondence Problem

$n_t \in \{1, ..., K\}$ is the index of the landmark perceived at time t

The variable n_t is often referred to as **correspondence**

Most theoretical work in SLAM assumes knowledge of correspondence

In practice, determining which landmark was observed is a **major challenge**

Traditional Approaches:

Maximum likelihood estimators for estimating correspondence on-the-fly

Only works well when landmarks are spaced sufficiently far apart

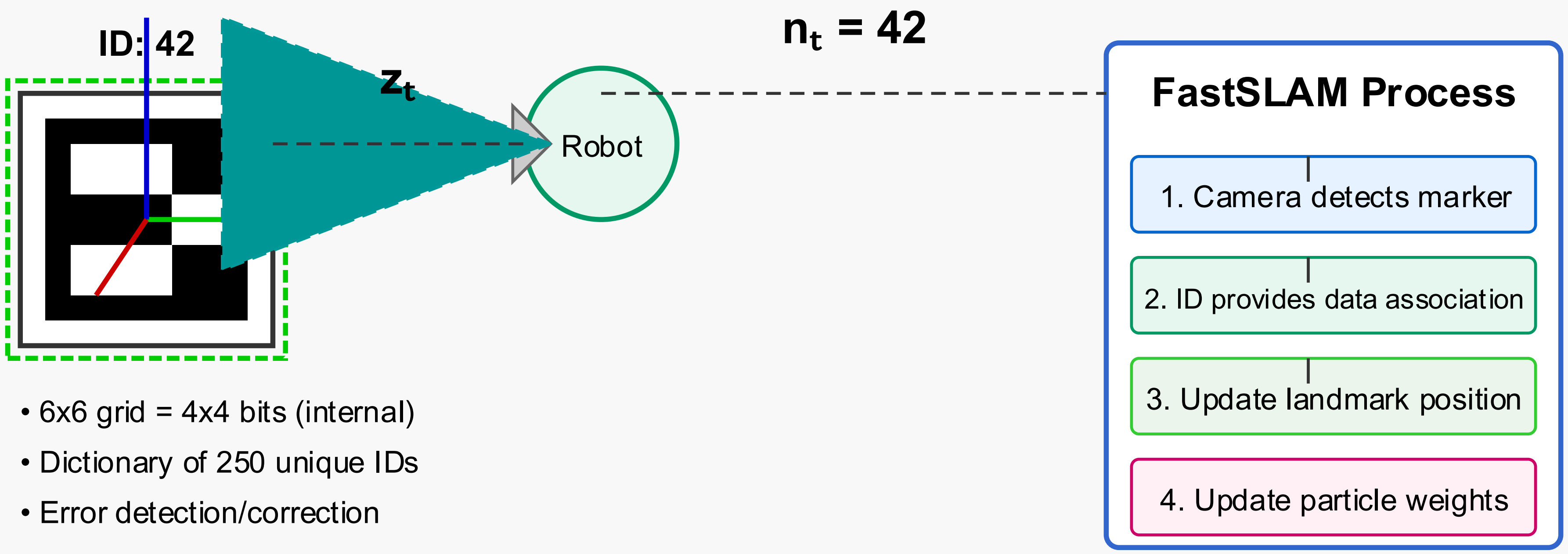
ArUco Marker Solution

- ArUco (Augmented Reality University of Cordoba) markers provide **uniquely identifiable landmarks**
- Each marker has a unique ID encoded in an n×n grid of black (0) and white (1) bits
- The correspondence problem **disappears** - we know *exactly* which landmark we're seeing
- SLAM problem simplifies to: $p(s_t, \theta \mid z_t, u_t, n_t)$

Key Advantage:

With known data associations, we can focus on the core SLAM problem without worrying about landmark identity confusion

ArUco Markers in FastSLAM Implementation



Benefits for FastSLAM Implementation

- Guaranteed correct data association (solves hardest SLAM problem)
- Precise landmark position and orientation measurement