SLAM: Simultaneous Localization and Mapping

Many slides courtesy of Ryan Eustice, Cyrill Stachniss, John Leonard

The SLAM Problem
A robot is exploring an unknown, static environment.

Given:
- The robot’s controls
- Observations of nearby features

Estimate:
- Map of features
- Path of the robot

SLAM Applications
- Indoors
- Undersea
- Space
- Underground

Illustration of SLAM without Landmarks
With only dead reckoning, vehicle pose uncertainty grows without bound
Illustration of SLAM without Landmarks

With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard
Illustration of SLAM without Landmarks

With only dead reckoning, vehicle pose uncertainty grows without bound

Copyright J. Leonard

Repeat, with Measurements of Landmarks

Illustration of SLAM with Landmarks

First position: two features observed

Second position: two new features observed

Copyright J. Leonard
Illustration of SLAM with Landmarks

Re-observation of first two features results in improved estimates for both vehicle and feature

Third position: two additional features added to map

Re-observation of first four features results in improved location estimates for vehicle and all features

Process continues as the vehicle moves through the environment

Courtesy J. Leonard
1. Move
2. Sense
3. Associate measurements with known features
4. Update state estimates for robot and previously mapped features
5. Find new features from unassociated measurements
6. Initialize new features
7. Repeat
Comparison with Ground Truth

Simultaneous Localization and Mapping (SLAM)
- Building a map and locating the robot in the map at the same time
- Chicken-and-egg problem

Definition of the SLAM Problem

Given
- The robot's controls
  \[ u_{1:T} = \{u_1, u_2, \ldots, u_T\} \]
- Observations
  \[ z_{1:T} = \{z_1, z_2, \ldots, z_T\} \]

Wanted
- Map of the environment
  \[ m \]
- Path of the robot
  \[ x_{0:T} = \{x_0, x_1, \ldots, x_T\} \]

Three Main Paradigms

Kalman filter
Particle filter
Graph-based
Bayes Filter

- Recursive filter with prediction and correction step

- Prediction
  \[ \tilde{p}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \tilde{p}(x_{t-1}) \, dx_{t-1} \]

- Correction
  \[ p(z_t \mid x_t) \tilde{p}(x_t) \]

EKF for Online SLAM

- We consider here the Kalman filter as a solution to the online SLAM problem

\[ p(x_t, m \mid z_{1:t}, u_{1:t}) \]

EKF SLAM

- Application of the EKF to SLAM
- Estimate robot’s pose and locations of landmarks in the environment
- Assumption: known correspondences
- State space (for the 2D plane) is

\[ x_t = (x_t, y_t, \theta, m_{1,x}, m_{1,y}, \ldots, m_{n,x}, m_{n,y})^T \]

EKF SLAM: State Representation

- Map with n landmarks: (3+2n)-dimensional Gaussian
- Belief is represented by

\[ (x_t, y_t, \theta, m_{1,x}, m_{1,y}, \ldots, m_{n,x}, m_{n,y}) \]
EKF SLAM: State Representation

More compactly

\[
\begin{pmatrix}
\sum_{R} x_R m_1 \\
\sum_{m_1} x_m m_1 \\
\vdots \\
\sum_{m_n} x_m m_n \\
\end{pmatrix}
\]

\[
\sum m_1 m_1 \\
\vdots \\
\sum m_n m_n \\
\]

EKF SLAM: State Representation

Even more compactly (note: \( x_R \rightarrow x \))

\[
\begin{pmatrix}
x \\
m \\
\end{pmatrix}
\]

\[
\sum x x \\
\sum m x \\
\sum m m \\
\]

EKF SLAM: Filter Cycle

1. State prediction
2. Measurement prediction
3. Measurement
4. Data association
5. Update

EKF SLAM: State Prediction

\[
\begin{pmatrix}
\sum_{R} x_R m_1 \\
\sum_{m_1} x_m m_1 \\
\vdots \\
\sum_{m_n} x_m m_n \\
\end{pmatrix}
\]

\[
\sum m_1 m_1 \\
\vdots \\
\sum m_n m_n \\
\]

Courtesy: Cyrill Stachniss
EKF SLAM: Measurement Prediction

\[
\begin{pmatrix}
\sum_{i=1}^{n} x_{i} \\
\sum_{i=1}^{n} m_{i}
\end{pmatrix}
\]

EKF SLAM: Obtained Measurement

\[
\begin{pmatrix}
\sum_{i=1}^{n} x_{i} \\
\sum_{i=1}^{n} m_{i}
\end{pmatrix}
\]

EKF SLAM: Data Association and Difference Between h(x) and z

\[
\begin{pmatrix}
\sum_{i=1}^{n} x_{i} \\
\sum_{i=1}^{n} m_{i}
\end{pmatrix}
\]

EKF SLAM: Update Step

\[
\begin{pmatrix}
\sum_{i=1}^{n} x_{i} \\
\sum_{i=1}^{n} m_{i}
\end{pmatrix}
\]
**EKF SLAM: Concrete Example**

**Setup**
- Robot moves in the 2D plane
- Velocity-based motion model
- Robot observes point landmarks
- Range-bearing sensor
- Known data association
- Known number of landmarks

**Initialization**
- Robot starts in its own reference frame (all landmarks unknown)
- 2N+3 dimensions
  \[
  \mu_0 = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & \infty \end{pmatrix}^T
  \]
  \[
  \Sigma_0 = \begin{pmatrix} 
  0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 & \infty \end{pmatrix}
  \]

**Extended Kalman Filter Algorithm**

1. **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. $\tilde{\mu}_t = \hat{y}(u_t, \mu_{t-1})$
3. $\tilde{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4. $K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1}$
5. $\mu_t = \tilde{\mu}_t + K_t(z_t - h(\tilde{\mu}_t))$
6. $\Sigma_t = (I - K_t H_t) \Sigma_t$
7. return $\mu_t, \Sigma_t$

**Prediction Step (Motion)**
- Goal: Update state space based on the robot’s motion
- Robot motion in the plane
  \[
  \begin{pmatrix} 
  x' \\ y' \\ \theta'
  \end{pmatrix} = \begin{pmatrix}
  x \\ y \\ \theta
  \end{pmatrix} + \begin{pmatrix}
  -\frac{\Delta t}{2} \sin \theta + \frac{\Delta t}{2} \sin(\theta + \omega \Delta t) \\ \frac{\Delta t}{2} \cos \theta - \frac{\Delta t}{2} \cos(\theta + \omega \Delta t) \\ \omega \Delta t\end{pmatrix}
  \]
  \[
  \begin{pmatrix}
  x' \\ y' \\ \theta'
  \end{pmatrix} = \begin{pmatrix}
  x \\ y \\ \theta
  \end{pmatrix} + \begin{pmatrix}
  -\frac{\Delta t}{2} \sin \theta + \frac{\Delta t}{2} \sin(\theta + \omega \Delta t) \\ \frac{\Delta t}{2} \cos \theta - \frac{\Delta t}{2} \cos(\theta + \omega \Delta t) \\ \omega \Delta t\end{pmatrix}
  \]
- How to map that to the 2N+3 dim space?
Update the State Space

- From the motion in the plane

\[
\begin{pmatrix}
\dot{x}' \\
\dot{y}' \\
\dot{\theta}'
\end{pmatrix}
= 
\begin{pmatrix}
x \\
y \\
\theta
\end{pmatrix}
+ 
\begin{pmatrix}
\frac{\Delta t}{2} \sin \theta + \frac{\Delta t}{2} \sin(\theta + \omega_1 \Delta t) \\
\frac{\Delta t}{2} \cos \theta - \frac{\Delta t}{2} \cos(\theta + \omega_1 \Delta t)
\end{pmatrix}
\]

- to the 2N+3 dimensional space

\[
\begin{pmatrix}
x' \\
y' \\
\theta'
\end{pmatrix}
= 
\begin{pmatrix}
x \\
y \\
\theta
\end{pmatrix}
+ 
\begin{pmatrix}
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0
\end{pmatrix}
\begin{pmatrix}
\frac{\Delta t}{2} \sin \theta + \frac{\Delta t}{2} \sin(\theta + \omega_1 \Delta t) \\
\frac{\Delta t}{2} \cos \theta - \frac{\Delta t}{2} \cos(\theta + \omega_1 \Delta t)
\end{pmatrix}
\]

\[
\left. \begin{array}{c}
\vdots \\
\vdots
\end{array} \right\}_{\text{3N+3}}
\]

Extended Kalman Filter Algorithm

1: Extended_Kalman_filter(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t):
2: \hat{\mu}_t = \hat{\theta}(u_t, \hat{z}_t) \quad \text{DONE}
3: \Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t
4: \hat{K}_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1}
5: \hat{\mu}_t = \hat{\mu}_t + \hat{K}_t(z_t - b(\hat{\mu}_t))
6: \Sigma_t = (I - \hat{K}_t H_t) \Sigma_t
7: return \mu_t, \Sigma_t

Update Covariance

- The function \( \hat{\theta} \) only affects the robot’s motion and not the landmarks

\[
G_t = \begin{pmatrix}
G_t^x & 0 \\
0 & I
\end{pmatrix}
\]

Identity (2N x 2N)

This Leads to the Time Propagation

1: Extended_Kalman_filter(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t):
2: \hat{\mu}_t = \hat{\theta}(u_t, \hat{z}_t) \quad \text{Apply & DONE}
3: \Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t
4: \Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t
5: \Sigma_t = \left( G_t^T \Sigma_{x} G_t + \Sigma_{y} \right) + R_t
6: return \mu_t, \Sigma_t
Extended Kalman Filter Algorithm

1:  \textbf{Extended Kalman filter}($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2:  \begin{align*}
\mu_t &= q(\mu_{t-1} + u_t) \text{ \textbf{DONE}} \\
\Sigma_t &= G_t \Sigma_{t-1} G_t^T + R_t \text{ \textbf{DONE}} \\
K_t &= \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1} \\
\mu_t &= \mu_t + K_t (z_t - h(\mu_t)) \\
\Sigma_t &= (I - K_t H_t) \Sigma_t \\
\text{return } \mu_t, \Sigma_t 
\end{align*}

EKF SLAM: Correction Step

- Known data association
- \( c_t^i = j \): \( i \)-th measurement at time \( t \) observes the landmark with index \( j \)
- Initialize landmark if unobserved
- Compute the expected observation
- Compute the Jacobian of \( h_t \)
- Proceed with computing the Kalman gain

Range-Bearing Observation

- Range-Bearing observation \( z_t^i = (r_t^i, \phi_t^i)^T \)
- If landmark has not been observed
  \[ \begin{pmatrix} \hat{\mu}_{j,x} \\ \hat{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \hat{\mu}_{t,x} \\ \hat{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \hat{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \hat{\mu}_{t,\theta}) \end{pmatrix} \]

Jacobian for the Observation

- Based on
  \[ \delta = \begin{pmatrix} \delta_x \\ \delta_y \\ \delta_{\hat{\mu}_{t,\theta}} \end{pmatrix} = \begin{pmatrix} \hat{\mu}_{j,x} - \hat{\mu}_{t,x} \\ \hat{\mu}_{j,y} - \hat{\mu}_{t,y} \\ -q \end{pmatrix} \]
  \[ q = \sqrt{q} \]
  \[ z_t^i = \begin{pmatrix} \delta_x \\ \delta_y \\ \delta_{\hat{\mu}_{t,\theta}} \end{pmatrix} \]

- Compute the Jacobian
  \[ \begin{pmatrix} \partial h(\hat{\mu}_t) \end{pmatrix} = \frac{\partial h(\hat{\mu}_t)}{\partial \mu_t} \]
  \[ = \frac{1}{q} \begin{pmatrix} -\sqrt{q} \delta_x & -\sqrt{q} \delta_y & 0 & +\sqrt{q} \delta_x & \sqrt{q} \delta_y \\ -\delta_x & -\delta_y & -q & -\delta_x & \delta_y \end{pmatrix} \]
Jacobian for the Observation

- Use the computed Jacobian

\[
\begin{align*}
\text{low } H_t^i &= \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_y & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_z & \sqrt{q}\delta_y \\
\delta_y & -\delta_z & -q & -\delta_y & \delta_z \\
\end{pmatrix} \\
\text{map it to the high dimensional space}
\end{align*}
\]

\[
H_t^i = \text{low } H_t^i F_{x,j} = 
\begin{pmatrix}
1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\
\end{pmatrix}
\]

Next Steps as Specified...

1: \text{Extended Kalman filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t):
2: \text{DONE}
3: \text{DONE}
4: \text{DONE}
5: \text{DONE}
6: \text{DONE}
7: \text{DONE}

Extended Kalman Filter Algorithm

1: \text{Extended Kalman filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t): 
2: \text{DONE}
3: \text{DONE}
4: \text{DONE}
5: \text{DONE}
6: \text{DONE}
7: \text{DONE}

EKF SLAM – Correction (1/2)

\[
\begin{align*}
Q_t &= \begin{pmatrix} \sigma_x^2 & 0 \\
0 & \sigma_y^2 \\
\end{pmatrix} \\
\text{for all observed features } z_i = (r_i, \phi_i)^T \text{ do} \\
j = c_i^j \\
\text{if landmark } j \text{ never seen before} \\
\delta_{j,x} = \frac{r_i^2 \cos(\phi_i + \hat{\mu}_{j,x})}{r_i^2 \sin(\phi_i + \hat{\mu}_{j,x})} \\
\delta_{j,y} = \frac{r_i^2 \sin(\phi_i + \hat{\mu}_{j,x})}{r_i^2 \sin(\phi_i + \hat{\mu}_{j,x})} \\
\text{endif} \\
\text{DONE} \\
\text{DONE} \\
\text{DONE} \\
\text{DONE} \\
\text{DONE}
\end{align*}
\]
EKF SLAM – Correction (2/2)

\[
F_{z,j} = \begin{pmatrix}
1 & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 0 & 0 & 0 & \ldots & 1 & 0 & \ldots & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & 1 & \ldots & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 1 \\
0 & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0
\end{pmatrix}
\]

\[H_j^i = \begin{pmatrix}
-\sqrt{q} \delta_z & -\sqrt{q} \delta_y & 0 & +\sqrt{q} \delta_x & +\sqrt{q} \delta_y & 0 & -\sqrt{q} \delta_y \\
-\sqrt{q} \delta_y & -\sqrt{q} \delta_z & -q & -\delta_x & +\sqrt{q} \delta_y & +\sqrt{q} \delta_y & 0 \\
\end{pmatrix}
\]

\[
K_j^i = \Sigma_{z} H_j^i (H_j^i \Sigma_{z} H_j^i + Q_j^i)^{-1}
\]

\[
\tilde{\mu}_j^n = \tilde{\mu}_j + K_j^i (c_j - \tilde{z}_j)
\]

\[
\Sigma_{\tilde{z}}_j = (I - K_j^i H_j^i) \Sigma_{z}
\]

\[
\text{return } \tilde{\mu}_j, \Sigma_{\tilde{z}}_j
\]

EKF SLAM Complexity

- Cubic complexity depends only on the measurement dimensionality
- Cost per step: dominated by the number of landmarks: \(O(n^2)\)
- Memory consumption: \(O(n^2)\)
- The EKF becomes computationally intractable for large maps!

Online SLAM Example

EKF SLAM Correlations

- In the limit, the landmark estimates become fully correlated
Approximate the SLAM posterior with a high-dimensional Gaussian \cite{Smith1986} …

Single hypothesis data association

In the real world, the mapping between observations and landmarks is unknown

Picking wrong data associations can have catastrophic consequences

EKF SLAM is brittle in this regard

Pose error correlates data associations

In loop-closing, data associations are under high ambiguity and possible environment symmetries.

Uncertainties collapse after a loop-closure (whether the closure was correct or not)
After the Loop-Closure

Example: Victoria Park Dataset

Victoria Park: Data Acquisition

Victoria Park: EKF Estimate
**Sub-maps for EKF SLAM**

![Sub-maps for EKF SLAM](Leonard_et_al_1998)

**EKF SLAM Summary**
- Quadratic in the number of landmarks: $O(n^2)$
- Convergence results for the linear case.
- Can diverge if nonlinearities are large!
- Have been applied successfully in large-scale environments.
- Approximations reduce the computational complexity.

**Literature**

**EKF SLAM**
- “Probabilistic Robotics”, Chapter 10
- Smith, Self, & Cheeseman: “Estimating Uncertain Spatial Relationships in Robotics”
- Dissanayake et al.: “A Solution to the Simultaneous Localization and Map Building (SLAM) Problem”
- Durrant-Whyte & Bailey: “SLAM Part 1” and “SLAM Part 2” tutorials

**Graph-SLAM**
- Full SLAM technique
- Generates probabilistic links
- Computes map only occasionally
- Based on Information Filter form
Graph-SLAM Idea

Information Form

• Represent posterior in canonical form
  \[ \Omega = \Sigma^{-1} \] Information matrix
  \[ \xi = \Sigma^{-1} \mu \] Information vector

• One-to-one transform between canonical and moment representation
  \[ \Sigma = \Omega^{-1} \]
  \[ \mu = \Omega^{-1} \xi \]

Information vs. Moment Form

Graph-SLAM Idea (1)
Efficient Map Recovery

- Information matrix inversion can be avoided if only best map estimate is required

- Minimize constraint function $J_{\text{GraphSLAM}}$ using standard optimization techniques (gradient descent, Levenberg Marquardt, conjugate gradient)
Robot Poses and Scans [Lu and Milios 1997]

- Successive robot poses connected by odometry
- Sensor readings yield constraints between poses
- Constraints represented by Gaussians
  \[ D_i = D_i + Q_{ij} \]
- Globally optimal estimate
  \[ \arg\max_x [P(D_i | D_{ij})] \]

Loop Closure

- Use scan patches to detect loop closure
- Add new position constraints
- Deform the network based on covariances of matches

Mapping the Allen Center

3D Outdoor Mapping

10^6 features, 10^5 poses, only few secs using cg.
Graph-SLAM Summary

- Addresses full SLAM problem
- Constructs link graph between poses and poses/landmarks
- Graph is sparse: number of edges linear in number of nodes
- Inference performed by building information matrix and vector (linearized form)
- Map recovered by reduction to robot poses, followed by conversion to moment representation, followed by estimation of landmark positions
- ML estimate by minimization of $J_{\text{GraphSLAM}}$
- Data association by iterative greedy search