Introduction to Mobile Robotics

SLAM:
Simultaneous Localization and Mapping
The SLAM Problem

SLAM is the process by which a robot builds a map of the environment and, at the same time, uses this map to compute its location.

- **Localization**: inferring location given a map
- **Mapping**: inferring a map given a location
- **SLAM**: learning a map and locating the robot simultaneously
The SLAM Problem

• SLAM is a *chicken-or-egg problem*:
  → A map is needed for localizing a robot
  → A good pose estimate is needed to build a map

• Thus, SLAM is regarded as a *hard problem* in robotics
The SLAM Problem

• SLAM is considered one of the most fundamental problems for robots to become truly autonomous.

• A variety of different approaches to address the SLAM problem have been presented

• Probabilistic methods rule!

• History of SLAM dates back to the mid-eighties (the stone-age of robotics)
The SLAM Problem

**Given:**

- The robot’s controls
  \[ U_{0:k} = \{u_1, u_2, \cdots, u_k\} \]
- Relative observations
  \[ Z_{0:k} = \{z_1, z_2, \cdots, z_k\} \]

**Wanted:**

- Map of features
  \[ m = \{m_1, m_2, \cdots, m_n\} \]
- Path of the robot
  \[ X_{0:k} = \{x_0, x_1, \cdots, x_k\} \]
The SLAM Problem

- **Absolute** robot pose
- **Absolute** landmark positions
- But only **relative** measurements of landmarks
Mapping with Raw Odometry

Video..
**SLAM Applications**

**SLAM is central** to a range of indoor, outdoor, in-air and underwater *applications* for both manned and autonomous vehicles.

**Examples:**

- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of abandoned mines
- Space: terrain mapping for localization
SLAM Applications

Indoors

Undersea

Space

Underground
Map Representations

Examples:
Subway map, city map, landmark-based map

Maps are **topological** and/or **metric models** of the environment
Map Representations

• Grid maps or scans, 2d, 3d

[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;...]

• Landmark-based

[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;...]
Why is SLAM a hard problem?

1. Robot path and map are both unknown

2. Errors in map and pose estimates correlated
Why is SLAM a hard problem?

- In the real world, the **mapping between observations and landmarks is unknown** (origin uncertainty of measurements)
- **Data Association**: picking **wrong** data associations can have **catastrophic** consequences (divergence)
SLAM: Simultaneous Localization And Mapping

- Full SLAM:

\[ p(x_{0:t}, m | z_{1:t}, u_{1:t}) \]

Estimates entire path and map!

- Online SLAM:

\[ p(x_t, m | z_{1:t}, u_{1:t}) = \int \int K \int p(x_{1:t}, m | z_{1:t}, u_{1:t}) \, dx_1 dx_2 \ldots dx_{t-1} \]

Integrations (marginalization) typically done recursively, one at a time

Estimates most recent pose and map!
Graphical Model of Full SLAM

\[ p(x_{1:t}, m | z_{1:t}, u_{1:t}) \]
Graphical Model of Online SLAM

\[
p(x_t, m | z_{1:t}, u_{1:t}) = \int \int K \int p(x_{1:t}, m | z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \ldots dx_{t-1}
\]
Graphical Model: Models

\[ x_k = f(x_{k-1}, u_k) \]

Motion model

\[ z_k = h(x_k, m) \]

Observation model
Remember? KF Algorithm

1. Algorithm **Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2. Prediction:

3. $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$

4. $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction:

6. $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$

7. $\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t)$

8. $\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$

9. Return $\mu_t$, $\Sigma_t$
EKF SLAM: State representation

- **Localization**
  
  3x1 pose vector
  3x3 cov. matrix
  
  \[
  x_k = \begin{bmatrix}
  x_k \\
  y_k \\
  \theta_k
  \end{bmatrix}, \quad
  C_k = \begin{bmatrix}
  \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\
  \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\
  \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta}^2
  \end{bmatrix}
  \]

- **SLAM**

  Landmarks are **simply added** to the state.
  **Growing** state vector and covariance matrix!

  \[
  x_k = \begin{bmatrix}
  x_R \\
  m_1 \\
  m_2 \\
  \vdots \\
  m_n
  \end{bmatrix}_k, \quad
  C_k = \begin{bmatrix}
  C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\
  C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\
  C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n}
  \end{bmatrix}_k
  \]
EKF SLAM: State representation

- Map with $n$ landmarks: $(3+2n)$-dimensional Gaussian

\[
\text{Bel}(x_t, m_t) = \begin{pmatrix}
x \\
y \\
\theta \\
l_1 \\
l_2 \\
M \\
l_N \\
\end{pmatrix},
\begin{pmatrix}
\sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\
\sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\
\sigma_{x\theta} & \sigma_{y\theta} & \sigma_{\theta}^2 \\
\sigma_{xl_1} & \sigma_{xl_1} & \sigma_{xl_2} & L & \sigma_{xl_N} \\
\sigma_{yl_1} & \sigma_{yl_1} & \sigma_{yl_2} & L & \sigma_{yl_N} \\
\sigma_{d_1} & \sigma_{d_1} & \sigma_{d_2} & L & \sigma_{d_N} \\
\sigma_{l_1l_1} & \sigma_{l_1l_2} & \sigma_{l_1l_2} & L & \sigma_{l_1l_N} \\
\sigma_{l_2l_2} & \sigma_{l_2l_2} & \sigma_{l_2l_2} & L & \sigma_{l_2l_N} \\
\sigma_{l_3l_3} & \sigma_{l_3l_3} & \sigma_{l_3l_3} & L & \sigma_{l_3l_N} \\
\end{pmatrix}
\]

- Can handle hundreds of dimensions
EKF SLAM: Building The Map

- State Prediction

Odometry:
\[
\hat{x}_R = f(x_R, u)
\]
\[
\hat{C}_R = F_x C_R F_x^T + F_u U F_u^T
\]

Robot-landmark cross-covariance prediction:
\[
\hat{C}_{RM_i} = F_x C_{RM_i}
\]
(skipping time index \(k\))
EKF SLAM: Building The Map

- Measurement Prediction

Global-to-local frame transform $h$

$$\hat{z}_k = h(\hat{x}_k)$$

$$x_k = \begin{bmatrix} x_R \\ m_1 \\ m_2 \\ \vdots \\ m_{mb} \end{bmatrix}_k$$

$$C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_{mb}R} & C_{M_{mb}M_1} & C_{M_{mb}M_2} & \cdots & C_{M_{mb}} \end{bmatrix}_k$$
EKF SLAM: Building The Map

- Observation

\( \mathbf{z}_k = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \)

\( \mathbf{R}_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \)

\[ \mathbf{x}_k = \begin{bmatrix} x_R \\ m_1 \\ m_2 \\ \vdots \\ m_{R_n} \end{bmatrix}_k \]

\[ \mathbf{C}_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k \]
EKF SLAM: Building The Map

- Data Association

Associates predicted measurements $\hat{z}_k^i$ with observation $z_k^j$

$$\nu_k^{ij} = z_k^j - \hat{z}_k^i$$

$$S_k^{ij} = R_k^j + H^i \hat{C}_k H^i T$$

(Gating)

$$x_k = \begin{bmatrix} x_R \\ m_1 \\ m_2 \\ \vdots \\ m_{n_2} \end{bmatrix}_k$$

$$C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1 R} & C_{M_1} & C_{M_1 M_2} & \cdots & C_{M_1 M_n} \\ C_{M_2 R} & C_{M_2 M_1} & C_{M_2} & \cdots & C_{M_2 M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_n R} & C_{M_n M_1} & C_{M_n M_2} & \cdots & C_{M_n} \end{bmatrix}_k$$
EKF SLAM: Building The Map

- Filter Update

The usual Kalman filter expressions

\[ K_k = \hat{C}_k H^T S_k^{-1} \]
\[ x_k = \hat{x}_k + K_k v_k \]
\[ C_k = (I - K_k H) \hat{C}_k \]

\[ x_k = \begin{bmatrix} x_R \\ m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix}_k \]

\[ C_k = \begin{bmatrix} C_{R} & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1 R} & C_{M_1} & C_{M_1 M_2} & \cdots & C_{M_1 M_n} \\ C_{M_2 R} & C_{M_2 M_1} & C_{M_2} & \cdots & C_{M_2 M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_n R} & C_{M_n M_1} & C_{M_n M_2} & \cdots & C_{M_n} \end{bmatrix}_k \]
Integrating New Landmarks

State augmented by

\[ m_{n+1} = g(x_R, z_j) \]
\[ C_{M_{n+1}} = G_R C_R G_R^T + G_z R_j G_z^T \]

landmark-to-landmark cross-covariance:

\[ C_{M_{n+1} M_i} = G_R C_{RM_i} \]
EKF SLAM

Map

Correlation matrix
EKF SLAM

Map

Correlation matrix
EKF SLAM

Map

Correlation matrix
KF-SLAM Properties (Linear Case)

- The **determinant** of any sub-matrix of the map covariance matrix **decreases monotonically** as successive observations are made.

- When a new landmark is initialized, its **uncertainty** is maximal.

- Landmark uncertainty **decreases monotonically** with each new observation.

[Dissanayake et al., 2001]
KF-SLAM Properties (Linear Case)

- In the limit, the landmark estimates become fully correlated

[Dissanayake et al., 2001]
KF-SLAM Properties (Linear Case)

- In the limit, the covariance associated with any single landmark location estimate is determined only by the initial covariance in the vehicle location estimate.

[Dissanayake et al., 2001]
Victoria Park Data Set

[courtesy by E. Nebot]
Victoria Park Data Set Vehicle

[courtesy by E. Nebot]
Data Acquisition

[courtesy by E. Nebot]
Estimated Trajectory

[courtesy by E. Nebot]
EKF SLAM Application

[courtesy by J. Leonard]
EKF SLAM Application

odometry

estimated trajectory

[courtesy by John Leonard]
SLAM Techniques for Generating Consistent Maps

• EKF SLAM

• FastSLAM (PF)

• Network-Based SLAM

• Hybrid Approaches (combination of NW+PF, NW+EKF)

• Topological SLAM (mainly place recognition)

• Scan Matching / Visual Odometry (only locally consistent maps)
EKF-SLAM: Complexity

- **Cost per step**: quadratic in \( n \), the number of landmarks: \( O(n^2) \)
- **Total cost** to build a map with \( n \) landmarks: \( O(n^3) \)
- **Memory**: \( O(n^2) \)

Approaches exist that make EKF-SLAM amortized 
\( O(n) / O(n^2) / O(n^2) \)

D&C SLAM [Paz et al., 2006]
EKF-SLAM: Summary

- **Convergence** for linear case!
- **Can diverge** if nonlinearities are large. And reality *is* nonlinear...
- Has been **applied successfully** in large-scale environments
- Approximations **reduce** the **computational complexity**
Data Association for SLAM

Interpretation tree

\[ S_{h_2} = \{ \{l_1, g_3\}, \{l_2, g_7\}, \{l_3, g_2\} \} \]
Data Association for SLAM

Env. Dyn.

\[ S_h = \{ \{l_1, g_4\}, \{l_2, g_8\}, \{l_3, *\} \} \]
Data Association for SLAM

Geometric Constraints

Location independent constraints

**Unary constraint:**
intrinsic property of feature
e.g. type, color, size

**Binary constraint:**
relative measure between features
e.g. relative position, angle

Location dependent constraints

**Rigidity constraint:**
"is the feature where I expect it given my position?"

**Visibility constraint:**
"is the feature visible from my position?"

**Extension constraint:**
"do the features overlap at my position?"

All decisions on a significance level $\alpha$
Data Association for SLAM

Interpretation Tree

[Grimson 1987], [Drumheller 1987], [Castellanos 1996], [Lim 2000]

Algorithm

- backtracking
- depth-first
- recursive
- uses geometric constraints
- exponential complexity
- absence of feature: no info.
- presence of feature: info. perhaps

```
function generate_hypotheses(h, L, G)
    H ← \{
    if \( L = \\{ \} \) then
        H ← H ∪ \{ h \}
    else
        i ← select_observation(L)
        for \( y \in G \) do
            p ← \( (L, g) \)
            if satisfy_unary_constraints(p) then
                if location_unavailable(h) then
                    accept ← satisfy_location_dependent_cnstr(L_h, p)
                    if accept then
                        h' ← h
                        \( S_{h'} \leftarrow S_h \cup \{ p \} \)
                        \( L_{h'} \leftarrow estimate_robot_location(S_{h'}) \)
                else
                    accept ← true
                    for \( p \in S_h \) while accept
                        accept ← satisfy_binary_constraints(p, p)
                    end
                    if accept then
                        h' ← h
                        \( S_{h'} \leftarrow S_h \cup \{ p \} \)
                        \( L_{h'} \leftarrow estimate_robot_location(S_{h'}) \)
                        if location_available(h') then
                            for \( p \in S_{h'} \) while accept
                                accept ← satisfy_location_dependent_cnstr(L_{h'}, p)
                            end
                        end
                    end
                    if accept then
                        generate_hypotheses(h', L \{ i \}, G)
                    end
                end
            end
        end
    return H
end
```
Data Association for SLAM

Pygmalion

\[ a = 0.95, \quad p = 2 \]
Data Association for SLAM

\[ a = 0.95, \quad p = 3 \]
Data Association for SLAM

\[ a = 0.95, \quad p = 4 \]

\[ a = 0.95, \quad p = 5 \]
Approximations for SLAM

• Local submaps
  [Leonard et al. 99, Bosse et al. 02, Newman et al. 03]

• Sparse links (correlations)
  [Lu & Milios 97, Guivant & Nebot 01]

• Sparse extended information filters
  [Frese et al. 01, Thrun et al. 02]

• Thin junction tree filters
  [Paskin 03]

• Rao-Blackwellisation (FastSLAM)
  [Murphy 99, Montemerlo et al. 02, Eliazar et al. 03, Haehnel et al. 03]