From Bayes’ rule to EKF SLAM

José Neira
Universidad de Zaragoza

jneira@unizar.es
http://www.cps.unizar.es/~jneira/
Outline

1. Introduction
2. The basic EKF SLAM algorithm
3. Feature extraction
4. Data Association
5. Perspective
Introduction: why EKFs?

• The first proposed solution for SLAM was based in the Extended Kalman Filter

• EKF is a very common tool in robotics

• It is helpful to analyze the properties of the SLAM problem

• Its wholesome stuff

P. Newman
Linear dynamic systems

- Dynamic model:

\[
p(x_{k+1} | x_k, u_k)
\]

\[
x_{k+1} = F_k x_k + B_k u_k + G_k v_k
\]

- \(x_k\): state vector of the system at instant \(t = kT\) (assumed to be \textbf{complete}, Markov assumption)
- \(u_k\): known control inputs
- \(v_k\): unknown system noise (due to system perturbations, model imperfections...)

2009 SLAM Summer School, Sydney
Sensor model

- Measurement model:

$$z_k = H_k x_k + w_k$$

- Probabilistic model

$$p(z_k | x_k)$$

$z_k$: measurement vector at instant $k$

$w_k$: unknown measurement noise (imprecision in the observation process and/or model)
Optimal Estimation

- Available information up to time $k$:

\[
Z^k = \{z_1, z_2, \ldots z_k\}
\]
\[
U^{k-1} = \{u_1, u_2, \ldots u_{k-1}\}
\]

- If state and measurement noises are white and independent, all information about the system is summarized in the conditional pdf:

\[
p_{k|k-1} \triangleq p(x_k|Z^{k-1}, U^{k-1}) \quad \text{Prediction}
\]
\[
p_k \triangleq p(x_k|Z^k, U^{k-1}) \quad \text{Estimation}
\]

- Optimal estimation (minimum mean squared error):

\[
\hat{x}_k^{MMSE} \triangleq \arg\min_{\hat{x}_k} E[(x_k - \hat{x}_k)^T (x_k - \hat{x}_k)|Z^k, U^{k-1}] = E[x_k|Z^k, U^{k-1}]
\]
Bayesian recursive estimation

\[ p_{k|k-1} = \int p(x_k|x_{k-1}, u_{k-1}) p_{k-1} \, dx_{k-1} \]

\[ p_k = \eta p(z_k|x_k) p_{k|k-1} \]

- No analytical solution in general

- Linear Gaussian case (**additive white Gaussian noise, AWGN**)

\[
\begin{align*}
    x_{k+1} &= F_k x_k + B_k u_k + G_k v_k \\
    z_k &= H_k x_k + w_k
\end{align*}
\]
Kalman filter

\[
p_{k|k-1} = \int p(x_k|x_{k-1}, u_{k-1}) \ p_{k-1} \ dx_{k-1}
\]

\[
p_k = \eta \ p(z_k|x_k) \ p_{k|k-1}
\]

\[
x_{k+1} = F_k x_k + B_k u_k + G_k v_k
\]

\[
z_k = H_k x_k + w_k
\]

- Prediction:

\[
\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_{k-1}
\]

\[
P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + G_k Q_k G_k^T
\]

- Update:

\[
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1})
\]

\[
P_{k|k} = (I - K_k H_k) P_{k|k-1}
\]

\[
K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}
\]

MMSE estimator

2009 SLAM Summer School, Sydney
Extended Kalman Filter

• Nonlinear dynamic systems

\[ x_{k+1} = f_k(x_k, u_k) + v_k \]
\[ z_k = h_k(x_k) + w_k \]

\[ F_k = \left. \frac{\partial f_k}{\partial x} \right|_{(\hat{x}_{k-1|k-1})} \]
\[ H_k = \left. \frac{\partial h_k}{\partial x} \right|_{(\hat{x}_{k|k-1})} \]

• Linear approximation of dynamic model and measurement model

• Prediction:

\[ \hat{x}_{k|k-1} = f_k(\hat{x}_{k-1|k-1}, u_{k-1}) \]
\[ P_{k|k-1} = F_kP_{k-1|k-1}F_k^T + G_kQ_kG_k^T \]

• Update:

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - h_k(\hat{x}_{k|k-1})) \]
\[ P_{k|k} = (I - K_kH_k)P_{k|k-1} \]
\[ K_k = P_{k|k-1}H_k^T \left( H_kP_{k|k-1}H_k^T + R_k \right)^{-1} \]

Best linear estimator
Outline

1. Introduction

2. The basic EKF SLAM algorithm

3. Feature extraction

4. Data Association

5. Perspective
My first EKF SLAM

- Environment information related to a set of elements:
  \[ \mathcal{F} = \{B, R, F_1, \ldots, F_n\} \]

- represented by a **stochastic map**: 
  \[ \mathcal{M}^B_m(\hat{x}^B, P^B) \]

\[ \begin{align*}
\hat{x}^B &= \begin{bmatrix}
\hat{x}^B_R \\
\vdots \\
\hat{x}^B_{F_n}
\end{bmatrix}, \\
P^B &= \begin{bmatrix}
P^B_{RR} & \ldots & P^B_{RF_n} \\
\vdots & \ddots & \vdots \\
P^B_{F_nR} & \ldots & P^B_{F_nF_n}
\end{bmatrix}
\end{align*} \]
My first EKF SLAM

Algorithm 1 SLAM:

\( x_0^B = 0; \ P_0^B = 0 \ \{ \text{Map initialization} \} \)

\([z_0, R_0] = \text{get\_measurements} \)

\([x_0^B, P_0^B] = \text{add\_new\_features}(x_0^B, P_0^B, z_0, R_0) \)

for \( k = 1 \) to steps do

\([x_{R_{k-1}}^R, Q_k] = \text{get\_odometry} \)

\([x_{k|k-1}^B, P_{k|k-1}^B] = \text{EKF\_prediction}(x_{k-1}^B, P_{k-1}^B, x_{R_k}^R, Q_k) \)

\([z_k, R_k] = \text{get\_measurements} \)

\( \mathcal{H}_k = \text{data\_association}(x_{k|k-1}^B, P_{k|k-1}^B, z_k, R_k) \)

\([x_k^B, P_k^B] = \text{EKF\_update}(x_{k|k-1}^B, P_{k|k-1}^B, z_k, R_k, \mathcal{H}_k) \)

\([x_k^B, P_k^B] = \text{add\_new\_features}(x_k^B, P_k^B, z_k, R_k, \mathcal{H}_k) \)

end for
Example: SLAM in a cloister

- Red dots: environment features (columns)
- Black line: robot trajectory
- Black semicircle: sensor range
Vehicle motion in 2D

\[
x^A_B = \begin{bmatrix} x_1 \\ y_1 \\ \phi_1 \end{bmatrix} \quad x^B_C = \begin{bmatrix} x_2 \\ y_2 \\ \phi_2 \end{bmatrix}
\]

Composition:

\[
x^A_C = x^A_B \oplus x^B_C = \begin{bmatrix} x_1 + x_2 \cos \phi_1 - y_2 \sin \phi_1 \\ y_1 + x_2 \sin \phi_1 + y_2 \cos \phi_1 \\ \phi_1 + \phi_2 \end{bmatrix}
\]

Inversion:

\[
x^B_A = \ominus x^A_B = \begin{bmatrix} -x_1 \cos \phi_1 - y_1 \sin \phi_1 \\ x_1 \sin \phi_1 - y_1 \cos \phi_1 \\ -\phi_1 \end{bmatrix}
\]
Odometry model:

\[
\begin{aligned}
\mathbf{x}^{R_{k-1}}_{R_k} &= \hat{\mathbf{x}}^{R_{k-1}}_{R_k} + \mathbf{v}_k \\
E[\mathbf{v}_k] &= 0 \\
E[\mathbf{v}_k\mathbf{v}_j^T] &= \delta_{k,j}Q_k
\end{aligned}
\]

Composition:

\[
\begin{aligned}
\hat{\mathbf{x}}^B_{R_k} &= \hat{\mathbf{x}}^B_{R_{k-1}} \oplus \hat{\mathbf{x}}^{R_{k-1}}_{R_k} \\
\mathbf{P}_{R_k} &\approx J_1 \mathbf{P}_{R_{k-1}} J_1^T + J_2 \mathbf{Q}_k J_2^T
\end{aligned}
\]

\[
\begin{aligned}
J_1 &= \frac{\partial \left( \mathbf{x}^B_{R_{k-1}} \oplus \mathbf{x}^{R_{k-1}}_{R_k} \right)}{\partial \mathbf{x}^B_{R_{k-1}}} \\
&\bigg|_{(\hat{\mathbf{x}}^B_{R_{k-1}}, \hat{\mathbf{x}}^{R_{k-1}}_{R_k})}

J_2 &= \frac{\partial \left( \mathbf{x}^B_{R_{k-1}} \oplus \mathbf{x}^{R_{k-1}}_{R_k} \right)}{\partial \mathbf{x}^{R_{k-1}}_{R_k}} \\
&\bigg|_{(\hat{\mathbf{x}}^B_{R_{k-1}}, \hat{\mathbf{x}}^{R_{k-1}}_{R_k})}
\end{aligned}
\]
Odometry in 2D

Jacobians:

\[ J_1 \oplus \{ x^A_B, x^B_C \} = \frac{\partial (x^A_B \oplus x^B_C)}{\partial x^A_B} \bigg|_{(\hat{x}^A_B, \hat{x}^B_C)} = \begin{bmatrix} 1 & 0 & -x_2 \sin \phi_1 - y_2 \cos \phi_1 \\ 0 & 1 & x_2 \cos \phi_1 - y_2 \sin \phi_1 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ J_2 \oplus \{ x^A_B, x^B_C \} = \frac{\partial (x^A_B \oplus x^B_C)}{\partial x^B_C} \bigg|_{(\hat{x}^A_B, \hat{x}^B_C)} = \begin{bmatrix} \cos \phi_1 & -\sin \phi_1 & 0 \\ \sin \phi_1 & \cos \phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ J_\ominus \{ x^A_B \} = \frac{\partial (\ominus x^A_B)}{\partial x^A_B} \bigg|_{(\hat{x}^A_B)} = \begin{bmatrix} -\cos \phi_1 & -\sin \phi_1 & -x_1 \sin \phi_1 - y_1 \cos \phi_1 \\ \sin \phi_1 & -\cos \phi_1 & x_1 \cos \phi_1 + y_1 \sin \phi_1 \\ 0 & 0 & -1 \end{bmatrix} \]
Odometry in 2D

How can we avoid unbounded uncertainty increase?
Odometry in 2D

- We assume that the odometry error follows a Gaussian distribution
Odometry in 2D

Vehicle error in theta (deg)
Map Features in 2D

\[
\mathbf{x}_P^B = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}
\]

Points:

\[
\mathbf{x}_P^A = \mathbf{x}_B^A \oplus \mathbf{x}_P^B = \begin{bmatrix} x_1 + x_2 \cos \phi_1 - y_2 \sin \phi_1 \\ y_1 + x_2 \sin \phi_1 + y_2 \cos \phi_1 \end{bmatrix}
\]

\[
J_1 \oplus \{ \mathbf{x}_B^A, \mathbf{x}_P^B \} = \begin{bmatrix} 1 & 0 & -x_2 \sin \phi_1 - y_2 \cos \phi_1 \\ 0 & 1 & x_2 \cos \phi_1 - y_2 \sin \phi_1 \end{bmatrix}
\]

Lines:

\[
\mathbf{x}_L^B = \begin{bmatrix} \rho_2 \\ \theta_2 \end{bmatrix}
\]

\[
\mathbf{x}_L^A = \mathbf{x}_B^A \oplus \mathbf{x}_L^B = \begin{bmatrix} x_1 \cos (\phi_1 + \theta_2) + y_1 \sin (\phi_1 + \theta_2) + \rho_2 \\ \phi_1 + \theta_2 \end{bmatrix}
\]

\[
J_1 \oplus \{ \mathbf{x}_B^A, \mathbf{x}_L^B \} = \begin{bmatrix} \cos (\phi_1 + \theta_2) & \sin (\phi_1 + \theta_2) & -x_1 \sin (\phi_1 + \theta_2) + y_1 \cos (\phi_1 + \theta_2) \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
J_2 \oplus \{ \mathbf{x}_B^A, \mathbf{x}_L^B \} = \begin{bmatrix} 1 & -x_1 \sin (\phi_1 + \theta_2) + y_1 \cos (\phi_1 + \theta_2) \\ 0 & 1 \end{bmatrix}
\]
Sensor measurements

- In polar coordinates:
  \[
  \hat{x} = \hat{d} \cos \hat{\phi} \\
  \hat{y} = \hat{d} \sin \hat{\phi}
  \]

- In cartesian coordinates:
  \[
  x = f(p) \\
  P_x \approx J P_p J^T \\
  J = \begin{bmatrix}
  \frac{\partial x}{\partial d} & \frac{\partial x}{\partial \phi} \\
  \frac{\partial y}{\partial d} & \frac{\partial y}{\partial \phi}
  \end{bmatrix}
  \]
  \[
  \hat{x} = (\hat{x}, \hat{y})^T \\
  P_x = \begin{bmatrix}
  \sigma^2_x & \sigma_{xy} \\
  \sigma_{xy} & \sigma^2_y
  \end{bmatrix}
  \]
The basic EKF SLAM Algorithm

OBSERVATIONS at step 1.6

Sensor measurements
EKF-SLAM: add new features

\[ \begin{align*}
\mathbf{x}_k^B &= \begin{pmatrix}
\mathbf{x}_{R_k}^B \\
\mathbf{x}_{F_{1,k}}^B \\
\vdots \\
\mathbf{x}_{F_{n,k}}^B
\end{pmatrix} \\
\Rightarrow \quad \mathbf{x}_{k+}^B &= \begin{pmatrix}
\mathbf{x}_{R_k}^B \\
\mathbf{x}_{F_{1,k}}^B \\
\vdots \\
\mathbf{x}_{F_{n,k}}^B \\
\mathbf{x}_{F_{n+1,k}}^B \\
\mathbf{x}_{R_k}^B \oplus \mathbf{z}_i
\end{pmatrix}
\end{align*} \]

Linearization:

\[ \begin{align*}
\mathbf{x}_{k+}^B &\simeq \tilde{\mathbf{x}}_{k+}^B + \mathbf{F}_k (\mathbf{x}_k^B - \tilde{\mathbf{x}}_k^B) + \mathbf{G}_k (\mathbf{z}_i - \hat{\mathbf{z}}_i) \\
\mathbf{P}_{k+}^B &= \mathbf{F}_k \mathbf{P}_k^B \mathbf{F}_k^T + \mathbf{G}_k \mathbf{R}_k \mathbf{G}_k^T
\end{align*} \]

Where:

\[ \begin{align*}
\Gamma_k &= \frac{\partial \mathbf{x}_{k+}^B}{\partial \mathbf{x}_k^B} = \begin{pmatrix}
1 & 0 & \cdots & 0 \\
0 & \cdots & \cdots & 0 \\
0 & 0 & \cdots & 1 \\
\mathbf{J}_1 \oplus \{\tilde{\mathbf{x}}_{R_k}^B, \tilde{\mathbf{z}}_i\} & 0 & \cdots & 0
\end{pmatrix} \\
\mathbf{G}_k &= \frac{\partial \mathbf{x}_{k+}^B}{\partial \mathbf{z}_i} = \begin{pmatrix}
0 \\
\vdots \\
0
\end{pmatrix}
\end{align*} \]
EKF-SLAM: add new features

\[
P_{k}^{B} = \begin{pmatrix}
P_{R} & P_{RF_1} & \cdots & P_{RF_n} \\
PT_{RF_1} & P_{F_1} & \cdots & P_{F_1F_n} \\
\vdots & \vdots & \ddots & \vdots \\
PT_{RF_n} & PT_{F_1F_n} & \cdots & P_{F_n}
\end{pmatrix}
\]

\[
P_{k+}^{B} = \begin{pmatrix}
P_{R} & P_{RF_1} & \cdots & P_{RF_n} & P_{RJ_{1+}^T} \\
PT_{RF_1} & P_{F_1} & \cdots & P_{F_1F_n} & P_{RF_1J_{1+}^T} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
PT_{RF_n} & PT_{F_1F_n} & \cdots & P_{F_n} & P_{RF_nJ_{1+}^T} \\
J_{1+}P_{R} & J_{1+}P_{RF_1} & \cdots & J_{1+}P_{RF_n} & J_{1+}P_{RJ_{1+}^T} + J_{2+}R_{k}J_{2+}^T
\end{pmatrix}
\]
EKF-SLAM: add new features
EKF-SLAM: compute robot motion

MAP at Step 2, features: 3, algorithm.
EKF-SLAM prediction: compute robot motion

\[ x^B_{R_k} = x^B_{R_{k-1}} \oplus x^R_{k-1} \]

Odometry model (white noise):

\[ x^R_{k-1} = \hat{x}^R_{k-1} + v_k \]
\[ E[v_k] = 0 \]
\[ E[v_kv^T_j] = \delta_{kj}Q_k \]

EKF prediction:

\[
\hat{x}^B_{k|k-1} = \begin{bmatrix}
\hat{x}^B_{R_{k-1}} \oplus \hat{x}^R_{k-1} \\
\hat{x}^B_{F_{1,k-1}} \\
\vdots \\
\hat{x}^B_{F_{m,k-1}}
\end{bmatrix}
\]

\[
F_k = \begin{bmatrix}
J_1 \ominus \left\{ \hat{x}^B_{R_{k-1}}, \hat{x}^R_{k-1} \right\} & 0 & \cdots & 0 \\
0 & I & \vdots \\
0 & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & I
\end{bmatrix}
\]

\[
P^B_{k|k-1} = F_k P^B_{k-1} F^T_k + G_k Q_k G^T_k
\]

\[
G_k = \begin{bmatrix}
J_2 \ominus \left\{ \hat{x}^B_{R_{k-1}}, \hat{x}^R_{k-1} \right\}
\end{bmatrix}
\]

2009 SLAM Summer School, Sydney
EKF SLAM: prediction

\[
\mathbf{P}_{k-1|k-1} = \begin{pmatrix}
\mathbf{P}_R & \mathbf{P}_{RF_1} & \ldots & \mathbf{P}_{RF_n} \\
\mathbf{P}^T_{RF_1} & \mathbf{P}_{F_1} & \ldots & \mathbf{P}_{F_1 F_n} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{P}^T_{RF_n} & \mathbf{P}^T_{F_1 F_n} & \ldots & \mathbf{P}_{F_n}
\end{pmatrix}
\]

\[
\mathbf{P}_{k|k-1} = \begin{pmatrix}
\mathbf{J}_1 \otimes \mathbf{P}_R \mathbf{J}^T_1 & + & \mathbf{J}_2 \otimes \mathbf{Q}_k \mathbf{J}^T_2 & & \mathbf{J}_1 \otimes \mathbf{P}_{RF_1} & \ldots & \mathbf{J}_1 \otimes \mathbf{P}_{RF_n} \\
\mathbf{J}^T_1 \otimes \mathbf{P}^T_{RF_1} & & & & \mathbf{P}_{F_1} & \ldots & \mathbf{P}_{F_1 F_n} \\
\vdots & & & & \vdots & \ddots & \vdots \\
\mathbf{J}^T_1 \otimes \mathbf{P}^T_{RF_n} & & & & \mathbf{P}^T_{F_1 F_n} & \ldots & \mathbf{P}_{F_n}
\end{pmatrix}
\]
EKF-SLAM: measurements

OBSERVATIONS at step 2

2009 SLAM Summer School, Sydney
EKF-SLAM: measurements

Observations at instant $k$:

$$z_{k,i} \quad \text{with } i = 1 \ldots s$$

Association Hypothesis (obs. $i$ with map feature $j_i$):

$$\mathcal{H}_k = [j_1, j_2, \ldots, j_s]$$

Measurement equation:

$$z_k = h_k(x_k^B) + w_k$$

Sensor model (white noise):

$$E[w_k] = 0$$

$$E[w_k w_j^T] = \delta_{k,j} R_k$$

$$E[w_k v_j^T] = 0$$
EKF-SLAM: predicted measurements

Linearization:

\[ z_k \sim h_k(\hat{x}^B_{k|k-1}) + H_k(x^B_k - \hat{x}^B_{k|k-1}) \]

\[ H_k = \left. \frac{\partial h_k}{\partial x^B_k} \right|_{(\hat{x}^B_{k|k-1})} = \left( H_R \ 0 \ \cdots \ H_F \ \cdots \ 0 \right) \]

\[ H_R = \left. \frac{\partial h_k}{\partial x^B_{R_k}} \right|_{(\hat{x}^B_{k|k-1})} ; \quad H_F = \left. \frac{\partial h_k}{\partial x^B_{F_k}} \right|_{(\hat{x}^B_{k|k-1})} \]
EKF-SLAM: Data association
**EKF-SLAM: map update**

State update:

\[
\hat{x}^B_k = \hat{x}^B_{k|k-1} + K_k (z_k - h_k(\hat{x}^B_{k|k-1}))
\]

Covariance update:

\[
P^B_k = (I - K_k H_k) P^B_{k|k-1}
\]

Filter gain:

\[
K_k = P^B_{k|k-1} H^T_k (H_k P^B_{k|k-1} H^T_k + R_k)^{-1}
\]
EKF-SLAM: map update

MAP at Step 2, features: 10, algorithm.
Why we do SLAM

Uncertainty still grows!
Moving forward…

Vehicle error in x (m)

Vehicle error in y (m)

Vehicle error in theta (deg)

$\sqrt{\det(P)} \times 10^{-3}$
Good news!

Loop closing reduces uncertainty!
Loop closing in EKF-SLAM

Vehicle error in x (m)

Vehicle error in y (m)

Vehicle error in theta (deg)

$\sqrt{\text{det}(P)}$ x $10^{-4}$

Loop closing reduces uncertainty!
Outline

1. Introduction

2. The basic EKF SLAM algorithm

3. Feature extraction

4. Data Association

5. Perspective
Laser segments

Split and merge:

1. Recursive Split:
   1. Obtain the line passing by the two extreme points
   2. Obtain the point more distant to the line
   3. If distance > error_max, split and repeat with the left and right sub-scan

2. Merge:
   1. If two consecutive segments are close enough, obtain the common line and the more distant point
   2. If distance <= error_max, merge both segments

3. Prune short segments

4. Estimate line equation

- Obtain line segments from a laser scan:
  - Segmentation
  - Line estimation
Split and Merge

Split

Split

Split

No more Splits

Merge

Split
Split and Merge

• Elimination of small segments:
  - Less than 6 pixels

• Segment fusion:
  - Between-segments distance < 10cm
Split and Merge

Not robust to complex and/or spurious data
RANSAC

$$(1 - w^n)^t = z$$

$t = \left[ \frac{\log z}{\log (1 - w^n)} \right]$

$p$ no. of points

$n$ points to build model

$w$ probability that a point is good

$O(p^n)$ possible models

$z$ acceptable probability of failure

$t$ tries

| w | 0,5 |
| n | 2   |
| z | 0,05 |
| t | 11  |
Robust statistics deal with spuriousness
RANSAC for 3D planes

Polaroid sonar

Very sparse and noisy data
Move and build a local map

Exploit redundancy

Use a good sensor model
Sonar Model for Points

Possible Point

Apparent Sonar Return

Hough Voting
for i in 1..n_positions
    for j in 1..n_sensors
        Compute $x^{Bi}_{Sj}$
        for $q^{Sj}_k$ in -$b$/2..$b$/2 step d
            Compute $q^{Bj}_k r^{Bj}_k$
            Vote($q^{Bj}_k$, $r^{Bj}_k$)
        end
    end
end

2009 SLAM Summer School, Sydney
Hough Transform: Corners

- Sonar returns **vote** for points
- Look for local maxima

**Hough Table for Points**

```
<table>
<thead>
<tr>
<th>Ro (m)</th>
<th>Theta (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>135</td>
</tr>
<tr>
<td>4</td>
<td>180</td>
</tr>
<tr>
<td>5</td>
<td>-45</td>
</tr>
</tbody>
</table>
```

**Point Groups**
Sonar Model for Lines

Hough Voting
for i in 1..n_positions
   for j in 1..n_sensors
      Compute $x_{S_j}^{Bi}$
      for $q_{S_j}^{Bi}$ in -$b$/2..$b$/2 step d
         Compute $q_{B_j}^{Bi}$, $r_{B_j}^{Bi}$
         Vote($q_{B_j}^{Bi}$, $r_{B_j}^{Bi}$)
      end
   end
end
Hough Transform: Lines

- Sonar returns **vote** for lines
- Look for local maxima

The Hough gives robust *local* data associations
Visual Features: monocular

- Interest point detectors: Harris, Shi-Tomasi, SIFT, SURF
- Monocular cameras: partial sensor information
Inverse depth representation

\[
\begin{pmatrix}
    x_c \\
    y_1 \\
    y_2 \\
    \ldots \\
    y_n \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
    x_i \\
    y_i \\
    z_i \\
    \theta_i \\
    \phi_i \\
    \rho_i \\
\end{pmatrix}
\]

- Camera position the first time the feature was seen
- Azimuth
- Elevation
- Inverse depth

Visual Features: stereo

Depth representation: 3D coordinates

\[ y_i = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]

Points can be initialized immediately with a single Gaussian
Depth vs. Inverse Depth
Outline

1. Introduction
2. The basic EKF SLAM algorithm
3. Feature extraction
4. Data Association
5. Perspective
The Data Association Problem

- $n$ map features: $\mathcal{F} = \{F_1 \ldots F_n\}$

- $m$ sensor measurements: $\mathcal{E} = \{E_1 \ldots E_m\}$

- Data association should return a hypothesis that associates each observation $E_i$ with a feature $F_{ji}$

$$\mathcal{H}_m = [j_1 \ldots j_i \ldots j_m]$$

- Non matched observations: $j_i = 0$
The Correspondence Space

Interpretation tree (Grimson et al. 87):

\((n + 1)^m\) possible hypotheses

Green points: measurements
Blue Points: predicted features
Why data association is difficult

- Low sensor error
- High sensor error
Why data association is difficult

- Low odometry error
- High odometry error
Why data association is difficult

- Low feature density
- High feature density
How important is data association?

A good algorithm

A bad algorithm
Why it’s difficult?

A good algorithm

A bad algorithm
Importance of Data Association

- EKF update:

\[
\hat{x}^B_k = \hat{x}^B_{k|k-1} + K_k \nu_k \\
P^B_k = (I - K_k H_k) P^B_{k|k-1} \\
K_k = P^B_{k|k-1} H_k^T (H_k P^B_{k|k-1} H_k^T + R_k)^{-1}
\]

- If the association of \( E_i \) with feature \( F_j \) is.....

  correct: \[ x - \hat{x} \]

  spurious: \[ \n \]

  error: \[ P \]

  covariance: \[ \]

  Consistency \[ \]

  Divergence!
**Individual Compatibility**

- Measurement equation for observation $E_i$ and feature $F_j$

  $$z_i = h_{ij}(x^B) + w_i$$

  $$z_i \approx h_{ij}(\hat{x}^B) + H_{ij}(x^B - \hat{x}^B)$$

- $E_i$ and $F_j$ are compatible if:

  $$D_{ij}^2 = (z_i - h_{ij}(\hat{x}^B))^T P_{ij}^{-1} (z_i - h_{ij}(\hat{x}^B)) < \chi^2_{d, \alpha}$$

  $$P_{ij} = H_{ij} P^B H_{ij}^T + R_i$$

  $$d = \text{length}(z_i)$$
The Fallacy of the Nearest Neighbor

Unrobust
Joint Compatibility

• Given a hypothesis $\mathcal{H} = [j_1, j_2, \ldots, j_s]$

• Joint measurement equation

$$z_\mathcal{H} = h_\mathcal{H}(x^B) + w_\mathcal{H}$$

$$h_\mathcal{H} = \begin{bmatrix}
h_{1j_1} \\
h_{2j_2} \\
\vdots \\
h_{sj_s}
\end{bmatrix}$$

• The joint hypothesis is compatible if:

$$D_{\mathcal{H}}^2 = (z_\mathcal{H} - h_\mathcal{H}(\hat{x}^B))^T C_{\mathcal{H}}^{-1} (z_\mathcal{H} - h_\mathcal{H}(\hat{x}^B)) < \chi^2_{d,\alpha}$$

$$C_{\mathcal{H}} = H_\mathcal{H} P^B H_\mathcal{H}^T + R_\mathcal{H}$$

$d = \text{length}(z)$
Joint Compatibility

Nearest neighbor vs. Joint Compatibility
Outline

1. Introduction
2. The basic EKF SLAM algorithm
3. Feature extraction
4. Data Association
5. Perspective
Perspective

• Basic EKF SLAM
  – Demonstration that a solution to SLAM is indeed possible.
  – It is the basis for many current successful SLAM algorithms in a wide range of applications.
  – $O(n^2)$ per step, it can run in real time for maps of size $n < 100$ features.

• Advanced EKF SLAM algorithms
  – Computational cost down to $O(n)$, $O(\log n)$?
Recommended readings


- **Divide and Conquer: EKF SLAM in \(O(n)\)**, L.M. Paz, J.D. Tardós, J. Neira, IEEE Transactions on Robotics Vol 24, Number 5, pp 1107-1120, October 2008


Bibliography
Data association

Innovation:

\[
\nu_k = z_k - \hat{z}_k \\
\text{Cov}(\nu_k) = H_k P_k B_k H_k^T + R_k^T
\]

Mahalanobis distance:

\[
D^2 = \nu_k^T \text{Cov}(\nu_k)^{-1} \nu_k \sim \chi_r^2
\]

where \( r = \text{dim}(\nu_k) \)

Hypothesis test:

\[
D^2 \leq \chi_{r, \alpha}^2 \Rightarrow z_k \text{ compatible with } \hat{z}_k
\]

where \( \alpha = 0.05 \) (common)
RANSAC

- Given a model that requires n data points to compute a solution and a set of data points P, with #(P) > n:
  - Randomly select a subset S1 of n data points and compute the model M1
  - Determine the consensus set S1* of points is P compatible with M1 (within some error tolerance)
  - If #(S1*) > th, use S1* to compute (maybe using least squares) a new model M1*
  - If #(S1*) < th, randomly select another subset S2 and repeat
  - If, after t trials there is no consensus set with th points, return with failure
The loop closing problem
Loop closing in mosaicing use first

Joint work with R. García, University of Girona
Loop closing in mosaicing: use Last

Sequential mosaicing is a form of odometry
The loop closing problem

- Loop beginning
- Loop end
The loop closing problem
The loop closing problem

Measurements (red) and predicted features (blue)
The loop closing problem

- Individual compatibility
- Joint Compatibility
The loop closing problem
Nearest Neighbor

Algorithm 2 Individual Compatibility Nearest Neighbor \( I\text{CNN} (E_{1\cdots m}, F_{1\cdots n}) \)

for \( i = 1 \) to \( m \) do \{measurement \( E_i \} \)

\[ D_{\text{min}}^2 \leftarrow \text{mahalanobis}^2(E_i, F_1) \]

\( \text{nearest} \leftarrow 1 \)

for \( j = 2 \) to \( n \) do \{feature \( F_j \} \)

\[ D_{ij}^2 \leftarrow \text{mahalanobis}^2(E_i, F_j) \]

if \( D_{ij}^2 < D_{\text{min}}^2 \) then

\( \text{nearest} \leftarrow j \)

\[ D_{\text{min}}^2 \leftarrow D_{ij}^2 \]

end if

end for

if \( D_{\text{min}}^2 \leq \chi^2_{d_i,1-\alpha} \) then

\( \mathcal{H}_i \leftarrow \text{nearest} \)

else

\( \mathcal{H}_i \leftarrow 0 \)

end if

end for

return \( \mathcal{H} \)

Greedy algorithm: \( O(mn) \)
Joint Compatibility Branch and Bound

- Find the largest hypothesis with **jointly consistent** pairings

```plaintext
procedure JCBB (H, i): -- find pairings for observation E_i

if i > m -- leaf node?
    if pairings(H) > pairings(Best)
        Best = H
    fi
else
    for j in {1...n}
        if individual_compatibility(i, j) and then
            joint_compatibility(H, i, j)
            JCBB([H j], i + 1) -- pairing (E_i, F_j) accepted
        fi
    rof
    if pairings(H) + m - i > pairings(Best) -- can do better?
        JCBB([H 0], i + 1) -- star node, E_i not paired
    fi
fi
```

Selects the largest set of pairings where there is **consensus**