Introduction to Mobile Robotics

Error Propagation, Feature Extraction, Extended Kalman Filter

Some slides adopted from: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Kai Arras and Probabilistic Robotics Book
Probabilistic robotics is fundamental for:
- **Representation**
- **Propagation**
- **Reduction**
  of uncertainty

**First-order error propagation** is fundamental for:
Kalman filter (KF), landmark extraction, KF-based localization and SLAM
Discrete Kalman Filter (review)

- Estimates the state $x$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

- with a measurement

$$z_t = C_t x_t + \delta_t$$

$A_t$ Matrix (nxn) that describes how the state evolves from $t$ to $t-1$ without controls or noise.

$B_t$ Matrix (nxl) that describes how the control $u_t$ changes the state from $t$ to $t-1$.

$C_t$ Matrix (kxn) that describes how to map the state $x_t$ to an observation $z_t$.

$\varepsilon_t$ Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance $R_t$ and $Q_t$ respectively.
Kalman Filter Algorithm

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

2. **Prediction:**
   
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$

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

5. **Correction:**
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \mu_t)$

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

9. **Return** $\mu_t, \Sigma_t$
The Prediction-Correction-Cycle

\begin{align*}
\overline{bel}(x_t) &= \begin{cases} 
\mu_t = a_t \mu_{t-1} + b_t u_t \\
\sigma_t^2 = a_t^2 \sigma_{t-1}^2 + \sigma_{act,t}^2
\end{cases} \\
\overline{bel}(x_t) &= \begin{cases} 
\mu_t = A_t \mu_{t-1} + B_t u_t \\
\Sigma_t = A_t \Sigma_{t-1} A_t^T + R_t
\end{cases}
\end{align*}
The Prediction-Correction-Cycle

\[
\begin{align*}
\text{bel}(x_t) &= \left\{ \begin{array}{l}
\mu_t &= \bar{\mu}_t + K_t (z_t - \bar{\mu}_t) \\
\sigma_t^2 &= (1 - K_t) \bar{\sigma}_t^2 \\
K_t &= \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \sigma_{\text{obs},t}^2}
\end{array} \right. \\
\Sigma_t &= (I - K_t C_t) \bar{\Sigma}_t
\end{align*}
\]

Correction
Gaussian Distribution

Why is the Gaussian distribution everywhere?

The importance of the normal distribution follows mainly from the **Central Limit Theorem**:

- The mean/sum of a large number of independent RVs, each with finite mean and variance (ergo not e.g. uniformly distributed RVs), will be approximately **normally distributed**.
- The more RVs the better the approximation.
Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

\[ x_t = g(u_t, x_{t-1}) \]

\[ z_t = h(x_t) \]

- Extended Kalman filter relaxes linearity assumption
First-Order Error Propagation

Approximating $f(X)$ by a **first-order** Taylor series expansion about the point $X = \mu_X$

$$Y \approx f(\mu_X) + \frac{\partial f}{\partial X} \bigg|_{X = \mu_X} (X - \mu_X)$$
Other Error Prop. Techniques

- **Second-Order Error Propagation**
  Rarely used (complex expressions)

- **Monte-Carlo**
  Non-parametric representation of uncertainties
  1. Sampling from $p(X)$
  2. Propagation of samples
  3. Histogramming
  4. Normalization
First-Order Error Propagation

X, Y assumed to be Gaussian

\[ Y = f(X) \]

Taylor series expansion

\[ Y \approx f(\mu_X) + \left. \frac{df}{dX} \right|_{X = \mu_X} (X - \mu_X) \]

Wanted: \( \mu_Y, \sigma_Y^2 \)  
(Solution on blackboard)
First-Order Error Propagation

\[ Y = f(X_1, X_2, \ldots, X_n) \]

Taylor series expansion (around mean)

\[ Y \approx f(\mu_1, \mu_2, \ldots, \mu_n) + \sum_{i=1}^{n} \left[ \frac{\partial f}{\partial X_i} (\mu_1, \mu_2, \ldots, \mu_n) \right] [X_i - \mu_i] \]

Wanted: \( \mu_Y, \sigma_Y^2 \) (Solution on blackboard)
First-Order Error Propagation

\[ Y = f(X_1, X_2, \ldots, X_n) \]
\[ Z = g(X_1, X_2, \ldots, X_n) \]

Wanted: \( \sigma_{YZ} \)

(Exercise)
First-Order Error Propagation

Putting things together...

\[
C_X = \begin{bmatrix}
\sigma_{X_1}^2 & \sigma_{X_1X_2} & \ldots & \sigma_{X_1X_n} \\
\sigma_{X_2X_1} & \sigma_{X_2}^2 & \ldots & \sigma_{X_2X_n} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{X_nX_1} & \sigma_{X_nX_2} & \ldots & \sigma_{X_n}^2 \\
\end{bmatrix}
\]

\[
C_Y = \begin{bmatrix}
\sigma_{Y_1}^2 & \sigma_{Y_1Y_2} \\
\sigma_{Y_2Y_1} & \sigma_{Y_2}^2 \\
\end{bmatrix}
\]

with

\[
\sigma_Y^2 = \sum_i \left( \frac{\partial f}{\partial X_i} \right)^2 \sigma_i^2 + \sum_{i \neq j} \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial f}{\partial X_j} \right) \sigma_{ij}
\]

\[
\sigma_{YZ}^2 = \sum_i \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial g}{\partial X_i} \right) \sigma_i^2 + \sum_{i \neq j} \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial g}{\partial X_j} \right) \sigma_{ij}
\]

→ “Is there a **compact form?**…”
Jacobian Matrix

• It’s a non-square matrix $n \times m$ in general

• Suppose you have a vector-valued function $f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}$

• Let the gradient operator be the vector of (first-order) partial derivatives

$$\nabla_x = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix}^T$$

• Then, the Jacobian matrix is defined as

$$F_x = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{bmatrix}$$
Jacobian Matrix

• It’s the orientation of the tangent plane to the vector-valued function at a given point

• Generalizes the gradient of a scalar valued function
• Heavily used for first-order error propagation...
First-Order Error Propagation

Putting things together...

\[ C_X = \begin{bmatrix}
\sigma_{x_1}^2 & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_n} \\
\sigma_{x_2x_1} & \sigma_{x_2}^2 & \cdots & \sigma_{x_2x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{x_nx_1} & \sigma_{x_nx_2} & \cdots & \sigma_{x_n}^2
\end{bmatrix} \]

\[ C_Y = \begin{bmatrix}
\sigma_{y_1}^2 & \sigma_{y_1y_2} \\
\sigma_{y_2y_1} & \sigma_{y_2}^2
\end{bmatrix} \]

with
\[ \sigma_Y^2 = \sum_i \left( \frac{\partial f}{\partial X_i} \right)^2 \sigma_i^2 + \sum_{i \neq j} \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial f}{\partial X_j} \right) \sigma_{ij} \]

\[ \sigma_{YZ}^2 = \sum_i \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial g}{\partial X_i} \right) \sigma_i^2 + \sum_{i \neq j} \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial g}{\partial X_j} \right) \sigma_{ij} \]

→ “Is there a compact form?...”
First-Order Error Propagation

...Yes! Given

- Input covariance matrix $C_X$
- Jacobian matrix $F_X$

the **Error Propagation Law**

\[
C_Y = F_X C_X F_X^T
\]

computes the output covariance matrix $C_Y$
First-Order Error Propagation

Alternative Derivation in Matrix Notation

\[ \mu_x = E(x) = E(Au + b) = AE(u) + b = A\mu_u + b \]

\[ \Sigma_x = E((x - E(x))(x - E(x))^T) = E((Au + b - AE(u) - b)(Au + b - AE(u) - b)^T) = E((A(u - E(u)))(A(u - E(u)))^T) = E((A(u - E(u)))(u - E(u))^T \, A^T)) = AE((u - E(u))(u - E(u))^T)A^T = A\Sigma_u A^T \]
Derivations (2/4)

Definitions

\[
\begin{align*}
\mu &= \mathbb{E}(X) \\
\text{Var}(X) &= \mathbb{E}((X - \mu)^2) \\
\text{Cov}(X, Y) &= \mathbb{E}((X - \mu)(Y - \nu))
\end{align*}
\]

Rules

\[
\begin{align*}
\mathbb{E}(X + c) &= \mathbb{E}(X) + c \\
\mathbb{E}(X + Y) &= \mathbb{E}(X) + \mathbb{E}(Y) \\
\mathbb{E}(aX) &= a\mathbb{E}(X)
\end{align*}
\]

Result SISO

\[
\begin{align*}
\mu_Y &= f(\mu_X), \\
\sigma_Y &= \left. \frac{\partial f}{\partial X} \right|_{X = \mu_X} \sigma_X.
\end{align*}
\]
Derivations (3/4)

Result MISO

\[ \mu_Y = E[Y] = E[a_0 + \sum_i a_i(X_i - \mu_i)] \]
\[ = E[a_0] + \sum_i E[a_iX_i] - E[a_i\mu_i] \]
\[ = a_0 + \sum_i a_iE[X_i] - a_iE[\mu_i] \]
\[ = a_0 + \sum_i a_i\mu_i - a_i\mu_i \]
\[ = a_0 \]

\[ \mu_Y = f(\mu_1, \mu_2, \ldots, \mu_n) \]

\[ \sigma_Y^2 = E[(Y - \mu_Y)^2] = E[(\sum_i a_i(X_i - \mu_i))^2] \]
\[ = E[\sum_i a_i(X_i - \mu_i)\sum_j a_j(X_j - \mu_j)] \]
\[ = E[\sum_i a_i^2(X_i - \mu_i)^2 + \sum_{i \neq j} a_i a_j(X_i - \mu_i)(X_j - \mu_j)] \]
\[ = \sum_i a_i^2 E[(X_i - \mu_i)^2] + \sum_{i \neq j} a_i a_j E[(X_i - \mu_i)(X_j - \mu_j)] \]
\[ = \sum_i a_i^2 \sigma_i^2 + \sum_{i \neq j} a_i a_j \sigma_{ij} \]

\[ \sigma_Y^2 = \sum_i \left( \frac{\partial f}{\partial X_i} \right)^2 \sigma_i^2 + \sum_{i \neq j} \left( \frac{\partial f}{\partial X_i} \right) \left( \frac{\partial f}{\partial X_j} \right) \sigma_{ij} \]
Derivations (4/4)

Result MIMO

$$\sigma_{YZ} = E[(Y - \mu_Y)(Z - \mu_Z)]$$

$$= E[Y \cdot Z] - E[Y]E[Z]$$

$$= E\left[\left(\mu_Y + \sum \frac{\partial f}{\partial X_i} [X_i - \mu_i]\right) \cdot \left(\mu_Z + \sum \frac{\partial g}{\partial X_i} [X_i - \mu_i]\right)\right] - \mu_Y \mu_Z$$

$$= E[\mu_Y \mu_Z + \mu_Z \sum \frac{\partial f}{\partial X_i} [X_i - \mu_i] + \mu_Y \sum \frac{\partial g}{\partial X_i} [X_i - \mu_i] + \sum \frac{\partial f}{\partial X_i} [X_i - \mu_i] \sum \frac{\partial g}{\partial X_i} [X_i - \mu_i]] - \mu_Y \mu_Z$$

$$= E[\mu_Y \mu_Z] + \mu_Z E\left[\sum \frac{\partial f}{\partial X_i} X_i - \sum \frac{\partial f}{\partial X_i} \mu_i\right] + \mu_Y E\left[\sum \frac{\partial g}{\partial X_i} X_i - \sum \frac{\partial g}{\partial X_i} \mu_i\right]$$

$$+ E\left[\sum \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_j} [X_i - \mu_i][X_j - \mu_j]\right] - \mu_Y \mu_Z$$

$$= \mu_Y \mu_Z + \mu_Z \sum \frac{\partial f}{\partial X_i} E[X_i] - \mu_Z \sum \frac{\partial f}{\partial X_i} E[\mu_i] + \mu_Y \sum \frac{\partial g}{\partial X_i} E[X_i] - \mu_Y \sum \frac{\partial g}{\partial X_i} E[\mu_i]$$

$$+ E\left[\sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_j} [X_i - \mu_i]^2 + \sum \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_j} [X_i - \mu_i][X_j - \mu_j]\right] - \mu_Y \mu_Z$$

$$= \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_i} E[(X_i - \mu_i)^2] + \sum \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_j} E[(X_i - \mu_i)(X_j - \mu_j)]$$

$$\sigma_{YZ} = \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_i} \sigma_i^2 + \sum \sum \frac{\partial f}{\partial X_i} \frac{\partial g}{\partial X_j} \sigma_{ij}^2$$
Feature Extraction: Motivation

**Landmarks** for:
- Localization
- SLAM
- Scene analysis

Examples:
- **Lines, corners, clusters:** good for indoor
- **Circles, rocks, plants:** good for outdoor
Wanted: Parameter Covariance Matrix

\[ C_{AR} = \begin{bmatrix} \sigma_A^2 & \sigma_{AR} \\ \sigma_{AR} & \sigma_R^2 \end{bmatrix} \]

Simplified sensor model:
all \( \sigma_{\theta_i}^2 = 0 \), independence

\[ C_{AR} = F_X C_X F_X^T \]

Result: Gaussians in the model space
Features: Properties

A feature/landmark is a **physical object** which is
- **static**
- **perceptible**
- (at least locally) **unique**

Abstraction from the raw data…
- **type** (range, image, vibration, etc.)
- **amount** (sparse or dense)
- **origin** (different sensors, map)

+ Compact, efficient, accurate, scales well, semantics
  – Not general
Feature Extraction

Can be subdivided into two subproblems:

- **Segmentation:** *Which* points contribute?
- **Fitting:** *How* do the points contribute?
Example: Local Map with Lines

Raw range data

Line segments
Example: Global Map with Lines

**Expo.02 map**
- 315 m²
- 44 Segments
- 8 kbytes
- 26 bytes / m²
- Localization accuracy ~1cm
Example: Global Map w. Circles

Victoria Park, Sydney

- Trees
Split and Merge

Picture by J. Tardos
Split and Merge

Algorithm

**Split**
- Obtain the line passing by the two extreme points
- Find the most distant point to the line
- If distance > threshold, split and repeat with the left and right point sets

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments
Split and Merge: Improvements

- Residual analysis before split

\[ \sum_{i = P_S} P_E d_i^2 > \sum_{i = P_S} P_B d_i^2 + \sum_{i = P_B} P_E d_i^2 \]

\( P_S, P_E, P_B \): start-, end-, break-point

Split only if the break point provides a "better interpretation" in terms of the error sum

[Castellanos 1998]
Split and Merge: Improvements

• Merge **non-consecutive** segments as a post-processing step

![Graphs showing split and merge improvements](image-url)
Line Representation

Choice of the line representation matters!

**Intercept-Slope**

\[ y = ax + b \]

\[ C = \begin{bmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ba} & \sigma_b^2 \end{bmatrix} \]

**Hessian model**

\[ x \cos \alpha + y \sin \alpha - r = 0 \]

\[ C' = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r \alpha} & \sigma_r^2 \end{bmatrix} \]

Each model has advantages and drawbacks
Least squares line fitting

- **Data:** \((x_1, y_1), \ldots, (x_n, y_n)\)
- **Line equation:** \(y_i = mx_i + b\)
- **Find** \((m, b)\) **to minimize**

\[
E = \sum_{i=1}^{n} (y_i - mx_i - b)^2
\]

\[
Y = \begin{bmatrix}
    y_1 \\
    \vdots \\
    y_n
\end{bmatrix}, \quad
X = \begin{bmatrix}
    x_1 & 1 \\
    \vdots & \vdots \\
    x_n & 1
\end{bmatrix}, \quad
B = \begin{bmatrix}
    m \\
    b
\end{bmatrix}
\]

\[
E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)
\]

\[
\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0
\]

\[
X^T XB = X^T Y
\]

**Normal equations:** least squares solution to \(XB = Y\)
Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines
Total least squares

- Distance between point \((x_i, y_i)\) and line \(ax + by = d\) \((a^2 + b^2 = 1)\): \(|ax_i + by_i - d|\)

Unit normal: \(N = (a, b)\)
Total least squares

- Distance between point \((x_i, y_i)\) and line \(ax + by = d\) \((a^2 + b^2 = 1)\): 
  \[|ax_i + by_i - d|\]
- Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]
Total least squares

- Distance between point \((x_i, y_i)\) and line \(ax+by=d (a^2+b^2=1): |ax_i + by_i - d|
- Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]

\[
\frac{dE}{dd} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0
\]

\[
E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)
\]

\[
d = \frac{a}{n} \sum_{i=1}^{n} x_i + \frac{b}{n} \sum_{i=1}^{n} y_i = a\bar{x} + b\bar{y}
\]

Solution to \((U^TU)N = 0\), subject to \(\|N\|^2 = 1\): eigenvector of \(U^TU\) associated with the smallest eigenvalue (least squares solution to homogeneous linear system \(UN = 0\))
Total least squares

\[ U = \begin{bmatrix}
    x_1 - \bar{x} & y_1 - \bar{y} \\
    \vdots & \vdots \\
    x_n - \bar{x} & y_n - \bar{y}
\end{bmatrix} \]

\[ U^T U = \begin{bmatrix}
    \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\
    \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2
\end{bmatrix} \]

second moment matrix
Total least squares

\[
U = \begin{bmatrix}
    x_1 - \bar{x} & y_1 - \bar{y} \\
    \vdots & \vdots \\
    x_n - \bar{x} & y_n - \bar{y}
\end{bmatrix}
\]

\[
U^T U = \begin{bmatrix}
    \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\
    \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2
\end{bmatrix}
\]

second moment matrix

\( N = (a, b) \)

\((x_i - \bar{x}, y_i - \bar{y})\)
Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line.

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}
\]

- Point on the line
- Noise: sampled from zero-mean Gaussian with std. dev. \( \sigma \)
- Normal direction

\[ax + by = d\]
Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix} = \begin{pmatrix}
  u \\
  v
\end{pmatrix} + \epsilon \begin{pmatrix}
  a \\
  b
\end{pmatrix}
\]

Likelihood of points given line parameters \((a, b, d)\):

\[
P(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = \prod_{i=1}^{n} P(x_i, y_i \mid a, b, d) \propto \prod_{i=1}^{n} \exp \left( -\frac{(ax_i + by_i - d)^2}{2\sigma^2} \right)
\]

Log-likelihood:

\[
L(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]
Line fitting

**Given:**
A set of $n$ points in polar coordinates

**Wanted:**
Line parameters

\[ \alpha \quad r \]

\[ \tan 2\alpha = \frac{2}{\Sigma w_i} \sum_{i < j} w_i w_j \rho_i \rho_j \sin(\theta_i + \theta_j) + \frac{1}{\Sigma w_i} \sum (w_i - \Sigma w_i) w_i \rho_i^2 \sin 2\theta_i \]

\[ r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i} \]

[Arras 1997]
LSQ Estimation

Regression, Least Squares-Fitting

\[ \epsilon_i = x_i \cos \alpha + y_i \sin \alpha - r \]

\[ S = \sum_{i=1}^{n} \epsilon_i^2 \]

Solve the non-linear equation system

\[ \frac{\partial S}{\partial \alpha} = 0 \quad \frac{\partial S}{\partial r} = 0 \]

Solution (for points in Cartesian coordinates):

→ Solution on blackboard
Circle Extraction

Can be formulated as a **linear** regression problem

Given $n$ points $\mathcal{P} = \{P_i\}_{i=1}^n$ with $P_i = (x_i\ y_i)^T$

Circle equation: $(x_i - x_c)^2 + (y_i - y_c)^2 = r_c^2$

Develop circle equation

\[
x_i^2 - 2x_ix_c + x_c^2 + y_i^2 - 2y_iy_c + y_c^2 = r_c^2
\]

\[
(-2x_i\ -2y_i\ 1) \begin{pmatrix} x_c \\ y_c \\ x_c^2 + y_c^2 - r_c^2 \end{pmatrix} = (-x_i^2\ -y_i^2)
\]

Parametrization trick
Circle Extraction

Leads to \textbf{overdetermined} equation system $A \cdot x = b$

$$A = \begin{pmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{pmatrix}, \quad b = \begin{pmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{pmatrix}$$

with vector of unknowns

$$x = (x_c \quad y_c \quad x_c^2 + y_c^2 - r_c^2)^T$$

Solution via \textbf{Pseudo-Inverse}

$$x = (A^T A)^{-1} A^T \cdot b$$

(assuming that $A$ has full rank)
Fitting Curves to Points

**Attention:** Always know the errors that you minimize!

**Algebraic** versus **geometric** fit solutions  

[Gander 1994]
LSQ Estimation: Uncertainties?

How does the **input uncertainty** propagate over the fit expressions to the **output**?

\[ X_1, \ldots, X_n : \text{Gaussian input random variables} \]

\[ A, R : \text{Gaussian output random variables} \]
Example: Line Extraction

**Wanted:** Parameter Covariance Matrix

\[
C_{AR} = \begin{bmatrix}
\sigma_A^2 & \sigma_{AR} \\
\sigma_{AR} & \sigma_R^2
\end{bmatrix}
\]

Simplified sensor model:
all \( \sigma_{\theta_i}^2 = 0 \), independence

\[
C_{AR} = F_X C_X F_X^T
\]

Result: Gaussians in the parameter space
Line Extraction in Real Time

- Robot *Pygmalion*
  EPFL, Lausanne
- CPU: PowerPC 604e at 300 MHz
  Sensor: 2 SICK LMS
- Line Extraction Times: $\sim 25$ ms
Derivations (1/4)

Result: Line Fit Cartesian Coordinates

(only for $r, \alpha$ more complicated...)

\[
\frac{\partial S}{\partial r} = 0
\]
\[
\Leftrightarrow \frac{\partial}{\partial r} \left\{ \sum \epsilon_i^2 \right\} = \sum \frac{\partial}{\partial r} \left\{ \epsilon_i^2 \right\} = 2 \sum \epsilon_i \frac{\partial}{\partial r} \left\{ \epsilon_i \right\}
\]
\[
\Leftrightarrow 2 \sum \left( x_i \cos \alpha + y_i \sin \alpha - r \right)(-1) = 0
\]
\[
\Leftrightarrow \sum \left( x_i \cos \alpha + y_i \sin \alpha - r \right) = 0
\]
\[
\Leftrightarrow \sum x_i \cos \alpha + \sum y_i \sin \alpha - nr = 0
\]
\[
\Leftrightarrow r = 1/n \sum x_i \cos \alpha + 1/n \sum y_i \sin \alpha
\]
\[
\Leftrightarrow r = \bar{x} \cos \alpha + \bar{y} \sin \alpha
\]
Introduction to Mobile Robotics

EKF Localization
Localization

“Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities.” [Cox ’91]

- **Given**
  - Map of the environment.
  - Sequence of sensor measurements.

- **Wanted**
  - Estimate of the robot’s position.

- **Problem classes**
  - Position tracking
  - Global localization
  - Kidnapped robot problem (recovery)
Landmark-based Localization

**EKF Localization:** Basic Cycle

![Diagram showing the basic cycle of EKF Localization with Prediction and Update steps]

---

55
Landmark-based Localization

**EKF Localization:** Basic Cycle

1. **Odometry or IMU** → **State Prediction** → **Update**
2. **Measurement Prediction**
3. **Data Association**
4. **Feature/Landmark Extraction** → **Sensors**
5. **Map**
Landmark-based Localization

**EKF Localization:** Basic Cycle

- **Encoder measurements**
  - Odometry or IMU

- **Predicted state**
  - State Prediction

- **Posterior state**
  - Update

- **Innovation from matched landmarks**

- **Predicted measurements in sensor coordinates**
  - Measurement Prediction

- **Landmarks in global coordinates**

- **Map**

- **Landmarks**

- **Raw sensory data**

- **Feature/Landmark Extraction**

- **Sensors**
Landmark-based Localization

**State Prediction** (Odometry)

\[ \hat{x}_k = f(x_{k-1}, u_k) \]
\[ \hat{C}_k = F_x C_k F_x^T + F_u U_k F_u^T \]

**Control** \( u_k \): wheel displacements \( s_l, s_r \)

\[ u_k = (s_l \ s_r)^T \]
\[ U_k = \begin{bmatrix}
\sigma_l^2 & 0 \\
0 & \sigma_r^2 
\end{bmatrix} \]

**Error model**: linear growth

\[ \sigma_l = k_l \ |s_l| \]
\[ \sigma_r = k_r \ |s_r| \]

**Nonlinear process model** \( f \):

\[
\begin{bmatrix}
x_k \\
y_k \\
\theta_k
\end{bmatrix} =
\begin{bmatrix}
x_{k-1} \\
y_{k-1} \\
\theta_{k-1}
\end{bmatrix} + \begin{bmatrix}
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l} \\
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l} \\
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l}
\end{bmatrix} \left( -\sin \theta_{k-1} + \sin \left( \theta_{k-1} + \frac{s_r-s_l}{b} \right) \right)
\]
\[
\begin{bmatrix}
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l} \\
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l} \\
\frac{b}{2} \frac{s_l+s_r}{s_r-s_l}
\end{bmatrix} \left( \cos \theta_{k-1} - \cos \left( \theta_{k-1} + \frac{s_r-s_l}{b} \right) \right)
\]
\[
\begin{bmatrix}
\frac{s_r-s_l}{b} \\
\frac{s_r-s_l}{b} \\
\frac{s_r-s_l}{b}
\end{bmatrix}
\]
Landmark-based Localization

State Prediction (Odometry)

\[ \hat{x}_k = f(x_{k-1}, u_k) \]
\[ \hat{C}_k = F_x C_k F_x^T + F_u U_k F_u^T \]

Control \( u_k \): wheel displacements \( s_l, s_r \)
\[ u_k = (s_l \ s_r)^T \]
\[ U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \]

Error model: linear growth
\[ \sigma_l = k_l |s_l| \]
\[ \sigma_r = k_r |s_r| \]

Nonlinear process model \( f \):
\[ x_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left( \sin \theta_{k-1} + \sin \left( \theta_{k-1} + \frac{s_r - s_l}{b} \right) \right) \\ \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left( \cos \theta_{k-1} - \cos \left( \theta_{k-1} + \frac{s_r - s_l}{b} \right) \right) \\ \frac{s_r - s_l}{b} \end{bmatrix} \]
Landmark-based Localization

**Landmark Extraction** (Observation)

Raw laser range data

Extracted lines

Extracted lines in model space

Hessian line model

\[ x \cos(\alpha) + y \sin(\alpha) - r = 0 \]
Landmark-based Localization

Measurement Prediction

- ...is a coordinate frame transform world-to-sensor
- Given the predicted state (robot pose), predicts the location $\hat{z}_k$ and location uncertainty $H\hat{C}_k H^T$ of expected observations in sensor coordinates

$$\hat{z}_k = h(\hat{x}_k, m)$$
Data Association (Matching)

- Associates predicted measurements $\hat{z}_k^i$ with observations $z_k^j$
  \[ \nu_{k}^{ij} = z_k^j - \hat{z}_k^i \]
  \[ S_k^{ij} = R_k^j + H^i \hat{C}_k H^i T \]

- Innovation $\nu_{k}^{ij}$ and innovation covariance $S_k^{ij}$

- Matching on significance level alpha

Green: observation
Magenta: measurement prediction

model space

No match!!
Wall was not observed.
Landmark-based Localization

Update

- Kalman gain

\[ K_k = \hat{C}_k H^T S_k^{-1} \]

- State update (robot pose)

\[ x_k = \hat{x}_k + K_k v_k \]

- State covariance update

\[ C_k = (I - K_k H) \hat{C}_k \]

Red: posterior estimate
Landmark-based Localization

- EKF Localization with Point Features
1. **EKF_localization** \( (\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m) \):

**Prediction:**

\[
G_t = \frac{\partial g(u_t, u_{t-1})}{\partial x_{t-1}} = \begin{pmatrix}
\frac{\partial x'}{\partial \mu_{t-1,x}} & \frac{\partial x'}{\partial \mu_{t-1,y}} & \frac{\partial x'}{\partial \mu_{t-1,\theta}} \\
\frac{\partial y'}{\partial \mu_{t-1,x}} & \frac{\partial y'}{\partial \mu_{t-1,y}} & \frac{\partial y'}{\partial \mu_{t-1,\theta}} \\
\frac{\partial \theta'}{\partial \mu_{t-1,x}} & \frac{\partial \theta'}{\partial \mu_{t-1,y}} & \frac{\partial \theta'}{\partial \mu_{t-1,\theta}} \\
\end{pmatrix}
\]

Jacobian of \( g \) w.r.t location

\[
B_t = \frac{\partial g(u_t, u_{t-1})}{\partial u_t} = \begin{pmatrix}
\frac{\partial x'}{\partial v_t} & \frac{\partial x'}{\partial \omega_t} \\
\frac{\partial y'}{\partial v_t} & \frac{\partial y'}{\partial \omega_t} \\
\frac{\partial \theta'}{\partial v_t} & \frac{\partial \theta'}{\partial \omega_t} \\
\end{pmatrix}
\]

Jacobian of \( g \) w.r.t control

\[
Q_t = \begin{pmatrix}
(\alpha_1 |v_t| + \alpha_2 |\omega_t|)^2 & 0 \\
0 & (\alpha_3 |v_t| + \alpha_4 |\omega_t|)^2 \\
\end{pmatrix}
\]

Motion noise

\[
\bar{\mu}_t = g(u_t, u_{t-1})
\]

Predicted mean

\[
\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + B_t Q_t B_t^T
\]

Predicted covariance
EKF Prediction Step
for different noise levels
left previous pose, right predicted pose
1. **EKF_localization** \( (\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m) \):

   **Correction:**

2. \[ \hat{z}_t = \left( \frac{\sqrt{(m_x - \mu_{t,x})^2 + (m_y - \mu_{t,y})^2}}{\text{atan}\,2(m_y - \mu_{t,y}, m_x - \mu_{t,x}) - \mu_{t,\theta}} \right) \] Predicted measurement mean

3. \[ H_t = \frac{\partial h(\mu_t, m)}{\partial x_t} = \left( \begin{array}{ccc} \frac{\partial r}{\partial \mu_{t,x}} & \frac{\partial r}{\partial \mu_{t,y}} & \frac{\partial r}{\partial \mu_{t,\theta}} \\ \frac{\partial \varphi}{\partial \mu_{t,x}} & \frac{\partial \varphi}{\partial \mu_{t,y}} & \frac{\partial \varphi}{\partial \mu_{t,\theta}} \end{array} \right) \] Jacobian of \( h \) w.r.t location

4. \[ R_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{pmatrix} \]

5. \[ S_t = H_t \bar{\Sigma}_t H_t^T + R_t \] Innovation covariance

6. \[ K_t = \bar{\Sigma}_t H_t^T S_t^{-1} \] Kalman gain

7. \[ \mu_t = \bar{\mu}_t + K_t (z_t - \hat{z}_t) \] Updated mean

8. \[ \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \] Updated covariance
EKF Observation Prediction Step
left column predictions – white circle ground truth
right measurement – predicted and observed
EKF Correction Step
left: given predicted and observed measurement, right corrected pose
Estimation Sequence (1)
solid line: ground truth, dashed line: odometry without sensing (left), with sensing (right)
Estimation Sequence (2)
solid line: ground truth, dashed line: odometry without sensing (left), with sensing (right)
Comparison to GroundTruth

solid line: ground truth, dashed line: odometry with sensor with different
EKF Localization Example

- [Arras et al. 98]: Laser range-finder and vision

High precision (<1cm accuracy)

Courtesy of K. Arras
EKF Localization Example

- Line and point landmarks
EKF Localization Example

- Line and point landmarks
EKF Localization Example

- **Expo.02**: Swiss National Exhibition 2002
- Pavilion "Robotics"
- 11 fully autonomous robots
- tour guides, entertainer, photographer
- 12 hours per day
- 7 days per week
- 5 months

- 3,316 km travel distance
- almost 700,000 visitors
- 400 visitors per hour

- Localization method: **Line-Based EKF**
EKF Localization Example

"Robotics"

Expo.02 Switzerland

May 15th - October 20th, 2002

Autonomous Systems Lab
EPFL

École Polytechnique Fédérale de Lausanne
Global EKF Localization

Interpretation tree

\[ S_{h_2} = \{ \{ l_1, g_3 \}, \{ l_2, g_7 \}, \{ l_3, g_2 \} \} \]
Global EKF Localization

Env. Dynamics

\[ S_h = \{ \{ l_1, g_4 \}, \{ l_2, g_8 \}, \{ l_3, \ast \} \} \]
Global EKF Localization

Geometric constraints we can exploit

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 $\alpha$
Global EKF Localization

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

Algorithm
• backtracking
• depth-first
• recursive
• uses geometric constraints

```
function generate_hypotheses(h, L, G)
    H ← {};
    if L = {} then
        H ← H ∪ {h};
    else
        l ← select_observation(L);
        for g ∈ G do
            p ← (l, g);
            if satisfy_unary_constraints(p) then
                if location_available(h) then
                    accept ← satisfy_location_dependent_cnstr(L, p);
                    if accept then
                        h' ← h;
                        S_h' ← S_h ∪ {p};
                        L_h' ← estimate_robot_location(S_h);
                    end
                else
                    accept ← true;
                    for p_p ∈ S_h while accept
                        accept ← satisfy_binary_constraints(p_p, p);
                    end
                    if accept then
                        h' ← h;
                        S_h' ← S_h ∪ {p};
                        L_h' ← estimate_robot_location(S_h);
                        if location_available(h) then
                            for p_p ∈ S_h while accept
                                accept ← satisfy_location_dependent_cnstr(L, p);
                            end
                        end
                    end
                end
            end
        end
    end
    generate_hypotheses(h', L \ {l}, G);
end
```

return H
Global EKF Localization

Pygmalion

\[ \alpha = 0.95, \quad p = 2 \]
Global EKF Localization

$\alpha = 0.95$, $p = 3$

Pygmalion
Global EKF Localization

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

Pyrgmalion
Global EKF Localization

\[ \alpha = 0.95 , \quad p = 5 \]

\[ t_{\text{exe}}: \textbf{633 ms} \] (PowerPC at 300 MHz)
Global EKF Localization
At Expo.02

05.07.02, 17.23 h

\( \alpha = 0.999 \)

[Arras et al. 03]
Global EKF Localization

At Expo.02

05.07.02, 17.23 h

\[ \alpha = 0.999 \]

\[ t_{exe} = 105 \text{ ms} \]

[Arras et al. 03]
Global EKF Localization

At Expo.02

05.07.02, 17.32 h

$\alpha = 0.999$

[Arras et al. 03]
Global EKF Localization

At Expo.02

\[ \alpha = 0.999 \]

\[ t_{exe} = 446 \text{ ms} \]

[Arras et al. 03]
EKF Localization Summary

- **EKF localization** implements *pose tracking*
- Very **efficient** and **accurate**
  (positioning error down to subcentimeter)
- Filter divergence can cause lost situations from which the EKF **cannot recover**
- Industrial applications

- **Global EKF localization** can be achieved using interpretation tree-based data association
- Worst-case complexity is **exponential**
- **Fast** in practice for **small** maps