Robotics - Localization - Kalman Filter & Extensions

Matteo Pirotta

pirotta@elet.polimi.it

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

23 May 2013

Inspired from Simone Ceriani’s slides (Robotics @ Como 2012)
Outline

1. Kalman Filter
2. K.F. Example
3. E.K.F.
4. EKF Loc. - Prediction
5. EKF Loc. - Update
6. Correspondences
7. U.K.F.
8. UKF Loc.
9. Considerations
Gaussian Distribution Reminder

**Multivariate Gaussian Distribution**

- $\mathbf{x}$ is a vector
- $\mathcal{N}(\mu, \Sigma)$, with
  - $\mu$: $n \times 1$, mean vector
  - $\Sigma$: $n \times n$, covariance matrix
- Linear transformation:
  - $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$
  - $\mathbf{Y} = A\mathbf{X} + \mathbf{b} \sim \mathcal{N}(A\mu + \mathbf{b}, A\Sigma A^T)$
- Product:
  - $\mathbf{X}_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$
  - $\mathbf{X}_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$
  - $\Pr(\mathbf{X}_1) \cdot \Pr(\mathbf{X}_2) \sim \mathcal{N} \left( \frac{\Sigma_2}{\Sigma_1 + \Sigma_2} \mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2} \mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}} \right)$
**Kalman Filter**

- Was introduced by R. E. Kálmán in 1960
- It is a technique for *optimal filtering* and *prediction* in *Linear Gaussian System*
- Implements belief computation
  - for continuous state
  - distributed as a multivariate Gaussian
  - i.e., belief is represented by $\mu$ and $\Sigma$
- The best studied technique for implementing Bayes Filters
Kalman Filter and Gaussian beliefs

- Hypothesis that guarantee that *posterior belief* is Gaussian

  1. **State transition probability** is a linear function with additive Gaussian noise:
     \[ x_t = A_t x_{t-1} + B_t u_t + \epsilon_t, \]
     where
     - \( x_t \) is the state vector \([n \times 1]\)
     - \( u_t \) is the control input \([m \times 1]\)
     - \( \epsilon_t \sim \mathcal{N}(0, R_t) \) is an \( n \)-dimensional Gaussian random variable that models the uncertainty
     - \( A_t \) is the state transition matrix \([n \times n]\)
     - \( B_t \) is the input transition matrix \([n \times m]\)
     - defines state transition probability \( \Pr(x_t | x_{t-1}, u_t) \)

  2. **Measurement probability** must be linear:
     \[ z_t = C_t x_t + \delta_t \]
     where
     - \( z_t \) is the measurement vector \([k \times 1]\)
     - \( C_t \) express the relation between state and measure \([k \times n]\)
     - \( \delta_t \sim \mathcal{N}(0, Q_t) \) is a \( k \)-dimensional Gaussian random variable that models the uncertainty

  3. **Initial belief** need to be normally distributed:
     \( \text{bel}(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0) \)
Discrete Kalman Filter - Algorithm

Algorithm Kalman filter\((\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\):

\[
\begin{align*}
\bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_T^T + R_t \\
K_t &= \bar{\Sigma}_t C_T (C_t \bar{\Sigma}_t C_T^T + Q_t)^{-1} \\
\mu_t &= \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\
\Sigma_t &= (I - K_t C_t) \bar{\Sigma}_t
\end{align*}
\]

return \(\mu_t, \Sigma_t\)

TWO STEP ALGORITHM

1. First Step - Prediction
   - Calculate the \(\text{bel}(x_t)\):
     \(\text{mean} (\bar{\mu})\) and covariance \(\bar{\Sigma}\)
   - Application of Gaussian properties

2. Second Step - Update
   - Calculate the \(\text{bel}(x_t)\):
     \(\text{mean} (\mu)\) and covariance \(\Sigma\)
   - Using the Kalman Gain \(K_t\) on the innovation, i.e. difference of
     - Measure: \(z_t\)
     - Expected Measure: \(C_t \bar{\mu}_t\)
Prediction Correction Cycle

\[
\text{bel}(x_t) = \begin{cases} 
\mu_t &= \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\
\Sigma_t &= (I - K_tC_t)\Sigma
\end{cases}, \quad K_t = \Sigma_t C_t^T \left( C_t^T \Sigma_t C_t + Q_t \right)^{-1}
\]

\[
\text{ber}(x_t) = \begin{cases} 
\bar{\mu}_t &= A_t\mu_{t-1} + B_t u_t \\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t
\end{cases}
\]
 Complexity

- **Highly efficient**: polynomial in measurement dimensionality $k$ and state dimensionality $n$:
  \[ O \left( k^{2.376} + n^2 \right) \]

- $O(k^{2.376})$ stands for matrix inversion (Coppersmith–Winograd algorithm)
- $O(n^2)$ due to matrix multiplication $K_t C_t$

- the efficiency of the KF is due to the fact that the parameters of the resulting Gaussian can be computed in closed form
**Falling body**

- Start at 0m with 0 m/s speed
- Standard deviation on position is 0[m]
- Standard deviation on speed is 0[m/s]
- A sensor measure the altitude in millimeters with a stochastic error of 100mm
- During the fall, stochastic errors affect
  - altitude, with std. dev. of 0.01m
  - speed, with std. dev. of 0.005m/s
Kalman Filter - Example - Definitions

**Definitions**
- **State**: \( x_t = [q_t, v_t]^T \), altitude and speed
- **Input**: \( u_t = g \), gravity
- **Constants**
  - \( b = 0.0025 \), friction coefficient
  - \( \Delta t = 0.001 \text{s}, \) period of discrete time step

**Initial State**
- \( \mu_0 = [0, 0]^T \), \( \Sigma_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \)

**State Transition**
- \( q_t = q_{t-1} + v_{t-1} \Delta t + \epsilon_{1t} \)
- \( v_t = v_{t-1} + g \Delta t - b v_{t-1} + \epsilon_{2t} \)
- **Matrix form:**
  \[
  \begin{bmatrix}
  q_t \\
  v_t
  \end{bmatrix} =
  \begin{bmatrix}
  1 & \Delta t \\
  0 & 1 - b
  \end{bmatrix}
  \begin{bmatrix}
  q_{t-1} \\
  v_{t-1}
  \end{bmatrix} +
  \begin{bmatrix}
  0 \\
  \Delta t
  \end{bmatrix}
  g
  +
  \begin{bmatrix}
  \epsilon_{1t} \\
  \epsilon_{2t}
  \end{bmatrix}
  \]

- \( R_t = \begin{bmatrix} 0.01^2 & 0 \\ 0 & 0.005^2 \end{bmatrix} \)
**ONLY PREDICTIONS EFFECTS**

- $\bar{\mu}_t = A\mu_{t-1} + Bu_t$
- $\Sigma_t = A\Sigma_{t-1}A^T + R$

**ALTITUDE PREDICTION**

Dashed lines are $\bar{\mu}_t \pm 3 \times \sqrt{\text{diag}(\Sigma)_t}$

**SPEED PREDICTION**
Kalman Filter - Example - Prediction vs True State

**What’s happen in the real world - State Transition**

**Altitude prediction vs true altitude**

**Speed prediction vs true speed**

Real Values is contained in the $3\sigma$ confidence interval
WHAT’S HAPPEN IN THE REAL WORLD - MEASUREMENT

SENSOR MEASUREMENT VS TRUE MEASURE
Kalman Filter - Example - Update Step

**Measurement Model**

- \( z_t = C_t x_t + \delta_t \)
- \( z_t = 1000 \cdot q_t + \delta_t \)
- \( C_t = \begin{bmatrix} 1000 & 0 \end{bmatrix} \)
- \( Q_t = \begin{bmatrix} 100^2 \end{bmatrix} \)

**The Update Step**

\[
\begin{align*}
K_t &= \tilde{\Sigma}_t C_t^T (C_t \tilde{\Sigma}_t C_t^T + Q_t)^{-1} \\
\mu_t &= \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\
\Sigma_t &= (I - K_t C_t) \tilde{\Sigma}_t
\end{align*}
\]

**An Intermediate Step**

- Let’s define
  - \( y_t = C_t \bar{\mu}_t \)
  - \( S_t = C_t \tilde{\Sigma}_t C_t^T + Q_t \)

- They represent a *measurement prediction*

- i.e., the (Gaussian) probability of the expected measurement \( \sim \mathcal{N}(y_t, S_t) \)
Kalman Filter - Example - Measurement Step

**MEASUREMENT PREDICTION**

Dashed lines are $y_t \pm 3 \times \sqrt{\text{diag}(S)_t}$

Up to now, no Update Step!!
Kalman Filter - Example - Run the update

**Prediction + Update**

**Altitude vs True Altitude**

The filter state tracks the real value!
Kalman Filter - Example - Comparisons

**Measurement Prediction**

**No Update**

![Graph showing measurement prediction without update](image)

**Measurement Prediction**

**With Update**

![Graph showing measurement prediction with update](image)
**Error on State**

- $\mathbf{x}_t^* - \mu_t$: difference between true value and estimated

**Altitude Error**

**Speed Error**
Outline

1. Kalman Filter
2. K.F. Example
3. E.K.F.
4. EKF Loc. - Prediction
5. EKF Loc. - Update
6. Correspondences
7. U.K.F.
8. UKF Loc.
9. Considerations
**Kalman Filter main issue**

- Works with linear systems
- Most real systems are modelled by nonlinear functions
- Can we “extend” it to treat non linear system?
- Yes, but under some hypothesis and with some drawbacks
- How to extend? → *local linearization*
  - A lot of linearization techniques
  - The extension of the KF exploits Taylor expansion
**GIVEN** \( x \sim \mathcal{N}(\mu, \sigma^2) \)

\[
y = ax + b
\]

\[
y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)
\]

The maintenance of a good Gaussian approximation depends on the shape of \( g(x) \).
Gaussian Variable Transformation - 2

Linearization via Taylor Expansion of $g(x)$

The maintenance of a good Gaussian approximation depends on the linearization point.
Linearization via Taylor Expansion of $g(x)$

Dependency of approximation quality on uncertainty

The maintenance of a good Gaussian approximation depends on the variance of $x$
**KF to EKF**

**KF**

**STATE TRANSITION**

\[ x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \]

**MEASUREMENT PROBABILITY**

\[ z_t = C_t x_t + \delta_t \]

**PREDICTION STEP**

\[ \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \]
\[ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \]

**UPDATE STEP**

\[ K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \]
\[ \mu_t = \bar{\mu}_t - K_t (z_t + C_t \bar{\mu}_t) \]
\[ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \]

**EKF**

**STATE TRANSITION**

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

**MEASUREMENT PROBABILITY**

\[ z_t = h(x_t, \delta_t) \]

**PREDICTION STEP**

\[ \bar{\mu}_t = g(\mu_{t-1}, u_t, 0) \]
\[ \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + N_t R_t N_t^T \]

**UPDATE STEP**

\[ K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + M_t Q_t M_t^T)^{-1} \]
\[ \mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t, 0)) \]
\[ \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \]
EKF - Jacobians

EKF
(non-additive noise formulation)

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

MEASUREMENT PROBABILITY
\[ z_t = h(x_t, \delta_t) \]

PREDICTION STEP
\[ \bar{\mu}_t = g(\mu_{t-1}, u_t, 0) \]
\[ \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + N_t R_t N_t^T \]

UPDATE STEP
\[ K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + M_t Q_t M_t^T)^{-1} \]
\[ \mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t, 0)) \]
\[ \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \]

JACOBIANS

- \[ G_t = \frac{\partial g(x, u, \epsilon)}{\partial x} \bigg|_{x=\mu_{t-1}, u=u_t, \epsilon=0} \]
derivative of the state transition function w.r.t. state variables

- \[ N_t = \frac{\partial g(x, u, \epsilon)}{\partial \epsilon} \bigg|_{x=\mu_{t-1}, u=u_t, \epsilon=0} \]
derivative of the state transition function w.r.t. noise variables

- \[ H_t = \frac{\partial h(x, \delta)}{\partial x} \bigg|_{x=\mu_{t-1}, \delta=0} \]
derivative of the measurement function w.r.t. state variables

- \[ M_t = \frac{\partial h(x, \delta)}{\partial \delta} \bigg|_{x=\mu_{t-1}, \delta=0} \]
derivative of the measurement function w.r.t. noise variables
EKF - Summary

**Complexity**

- Highly Efficient: polynomial in measurement dimension $k$ and state dimension $n$

\[ O\left(k^{2.376} + n^2\right) \]

- Same complexity of KF

- Not optimal. May diverge if nonlinearities are large.

**Extension**

- Multiple distinct hypotheses (MHEKF algorithm)
  - Multi-modal representation for the posterior belief
  - e.g., mixtures of Gaussians

\[
\text{bel}(x_t) = \frac{1}{\sum_j \phi_{t,j}} \sum_j \phi_{t,j} \text{det} \left(2\pi \Sigma_{j,t}\right)^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (x_t - \mu_{j,t})^T \Sigma_{j,t}^{-1} (x_t - \mu_{j,t})\right)
\]

where $\phi_{t,j} > 0$

- Use methods different from Taylor’s expansion to perform linearization
  - e.g., unscented Kalman filter, moments matching
Outline

1. Kalman Filter
2. K.F. Example
3. E.K.F.
4. EKF Loc. - Prediction
5. EKF Loc. - Update
6. Correspondences
7. U.K.F.
8. UKF Loc.
9. Considerations
Localization with E.K.F.

**Localization**
- Special case of Markov Localization
- EKF can treat the *pose tracking* problem
- We can consider *uncertainty* (or *belief*) locally Gaussian

**State transition - Prediction**
- The next robot position using motion information

**Measurement - Update**
- Sense of a *map landmark*
  \[ z_t = \{ z_t^1, z_t^2, \ldots \} \]
  e.g., sense distance and angle

**Map and correspondences**
- The map is a collection of features
- Position of landmarks in world coordinates
- Landmarks are uniquely identifiable e.g., different colors
- Identify an unique correspondence \( c_t \) to \( z_t \) (\( z_t \leftrightarrow c_t \))
Localization with E.K.F. – Example

**CORRESPONDENCES**

\[ z_t \leftrightarrow c_t \]

\[ c_t = \{ \text{door}_1, \text{door}_2, \text{door}_3 \} \]

**BEHAVIOR**

1. Initial belief is placed near Door 1
2. The robots moves to the right, its belief is convolved with the motion model
3. The robot senses Door 2 \((z_t \leftrightarrow c_t = \text{door}_2)\). The incorporation of the measurement into robot belief generates the posterior
Localization with E.K.F. – Idea

**Given**
- Robot pose at time $t - 1$ ($\sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$)
- Control $u_t$
- Map $m$
- Feature correspondences $z_t \leftrightarrow c_t$

**Output**
- $x_t \sim \mathcal{N}(\mu_t, \Sigma_t)$
- Likelihood $Pr_{z_t}$
**STATE**

- The robot position and orientation (2D)
  - \([x, y]\): the robot position
  - \(\theta\): orientation

  in world reference frame: \(T_W^{R_t}\)

**STATE PREDICTION**

- Input \(u\) : \([\Delta x, \Delta y, \Delta \theta] = [v_x \Delta t, v_y \Delta t, \omega \Delta t]\)
- Relative motion: \(T_{R_{t+1}}^{R_t}(u)\)
- Prediction: \(T_W^{R_{t+1}} = T_W^{R_t} \cdot T_{R_{t+1}}^{R_t}\)

\[
\begin{bmatrix}
\cos(\theta_t) & -\sin(\theta_t) & x_t \\
\sin(\theta_t) & \cos(\theta_t) & y_t \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\Delta \theta) & -\sin(\Delta \theta) & \Delta x \\
\sin(\Delta \theta) & \cos(\Delta \theta) & \Delta y \\
0 & 0 & 1
\end{bmatrix}
= \\
\begin{bmatrix}
\cos(\theta_t + \Delta \theta) & -\sin(\theta_t + \Delta \theta) & \cos(\theta_t) \Delta x - \sin(\theta_t) \Delta y + x_t \\
\sin(\theta_t + \Delta \theta) & \cos(\theta_t + \Delta \theta) & \sin(\theta_t) \Delta x + \cos(\theta_t) \Delta y + y_t \\
0 & 0 & 1
\end{bmatrix}
\]
STATE PREDICTION

- \( x_t = g(x_{t-1}, u_t, 0) \) (no noise)
  \[
  \begin{align*}
  x_{t+1} &= \cos(\theta_t) \Delta x - \sin(\theta_t) \Delta y + x_t \\
  y_{t+1} &= \sin(\theta_t) \Delta x + \cos(\theta_t) \Delta y + y_t \\
  \theta_{t+1} &= \theta_t + \Delta \theta
  \end{align*}
  \]

NOISE INTRODUCTION

- Suppose that noise affects inputs
  \[
  \begin{align*}
  \tilde{\Delta} x &= \Delta x + \epsilon_x \\
  \tilde{\Delta} y &= \Delta y + \epsilon_y \\
  \tilde{\Delta} \theta &= \Delta \theta + \epsilon_\theta
  \end{align*}
  \]
- \( \epsilon = [\epsilon_x, \epsilon_y, \epsilon_\theta] \sim \mathcal{N}(0, \Sigma_\epsilon) \)

- \( x_{t+1} = \cos(\theta_t) \tilde{\Delta} x - \sin(\theta_t) \tilde{\Delta} y + x_t \)
- \( y_{t+1} = \sin(\theta_t) \tilde{\Delta} x + \cos(\theta_t) \tilde{\Delta} y + y_t \)
- \( \theta_{t+1} = \tilde{\Delta} \theta + \theta_t \)

JACOBIANS

- \( G_t = \left. \frac{\partial g(x,u,\epsilon)}{\partial x} \right|_{x=\mu_{t-1},u=u_t,\epsilon=0} \)
- \[
  G_t = \begin{bmatrix}
  \frac{\partial g_1(x,u,\epsilon)}{\partial x} & \frac{\partial g_1(x,u,\epsilon)}{\partial y} & \frac{\partial g_1(x,u,\epsilon)}{\partial \theta} \\
  \frac{\partial g_2(x,u,\epsilon)}{\partial x} & \frac{\partial g_2(x,u,\epsilon)}{\partial y} & \frac{\partial g_2(x,u,\epsilon)}{\partial \theta} \\
  \frac{\partial g_3(x,u,\epsilon)}{\partial x} & \frac{\partial g_3(x,u,\epsilon)}{\partial y} & \frac{\partial g_3(x,u,\epsilon)}{\partial \theta}
  \end{bmatrix}
  \]
- \( \frac{\partial g_1(x,u,\epsilon)}{\partial x} = 1, \quad \frac{\partial g_1(x,u,\epsilon)}{\partial y} = 0 \)
- \( \frac{\partial g_1(x,u,\epsilon)}{\partial \theta} = -\sin(\theta) \Delta x - \cos(\theta) \Delta y \)
- \[
  N_t = \left. \frac{\partial g(x,u,\epsilon)}{\partial \epsilon} \right|_{x=\mu_{t-1},u=u_t,\epsilon=0} \]
- \[
  N_t = \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) & 0 \\
  \sin(\theta) & \cos(\theta) & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
Initial state

**EKF STATE**

- \( \mu_0 = [0, 0, 0] \)
- \( \Sigma_0 = \begin{bmatrix} 0.1^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0 & \text{deg2rad}(10)^2 \end{bmatrix} \)

- Robot is in the origin
- But we have some uncertainty on its position and orientation
- e.g., Robot real pose is \([0.094, 0.069, 5.2769^\circ]\)
- \( \mathbf{C} = k \cdot \Sigma_0^{-1} \) confidence ellipse

blue: true position
red: estimated position \((\mu)\)
a (noisy) input arrives, the prediction step is performed

\[
\overline{\mu}_t = g(\mu_{t-1}, u_t, 0) \\
\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + N_t R_t N_t^T
\]

**blue**: true position

**red**: estimated position (\(\mu\))
After some steps...

\[ \mu_t = g(\mu_{t-1}, u_t, 0) \]
\[ \Sigma_t = G_t \Sigma_{t-1} G_t^T + N_t R_t N_t^T \]

- blue: true position and trajectory
- red: estimated position and trajectory \((\mu_t)\)
Only Prediction - 3

The complete path with the prediction step
Notice that the true position is inside the ellipse
Only Prediction - Graphs

**Complete execution**

The complete path with the prediction step
Notice that the true position is inside the ellipse
A snapshot of code

\[
G = \begin{bmatrix}
1, & 0, & -dy*cos(th) - dx*sin(th); \\
0, & 1, & dx*cos(th) - dy*sin(th); \\
0, & 0, & 1
\end{bmatrix};
\]

\[
N = \begin{bmatrix}
cos(th), & -sin(th), & 0; \\
sin(th), & cos(th), & 0; \\
0, & 0, & 1
\end{bmatrix};
\]

\[
x1 = \cos(th)^*(dx) - \sin(th)^*(dy) + x;
\]

\[
y1 = \sin(th)^*(dx) + \cos(th)^*(dy) + y;
\]

\[
\text{th1} = \text{th} + \text{dth};
\]

\[
\text{ekf.mu} = [x1,y1,\text{th1}]';
\]

\[
\text{ekf.sigma} = G*\text{ekf.sigma} \ast G' + N \ast R \ast N';
\]
Outline

1. Kalman Filter
2. K.F. Example
3. E.K.F.
4. EKF Loc. - Prediction
5. EKF Loc. - Update
6. Correspondences
7. U.K.F.
8. UKF Loc.
9. Considerations
**The Map**

- \( m : \{ p_{1}^{(W)}, p_{2}^{(W)}, \ldots, p_{m}^{(W)} \} \)
- i.e., a set of points in world coordinates
- Known with *absolute* precision
**The Sensor**

- Measure points in *polar coordinates*
  
  i.e., \( \rho, \theta \) values

- w.r.t. robot reference frame

- It recognize the ID of the landmark
  
  i.e., Landmarks uniquely identifiable
  
  Correspondences are known
  
  No data association issues

- Physical limits:
  
  - Min and max distance
  
  - Min and max angle
  
  - Additive zero mean noise on measures
    
    both for distance and angle
**Measurement & Update Step - The equation**

**Measurement**

- \( \mathbf{x} = [x, y, \theta] \) is the EKF state
  - i.e., the robot complete pose
- Measure: \( h_i(\mathbf{x}, \mathbf{p}^{(W)}_i, \delta) \)
  - It expresses what we expect from the sensor
  - Given a single map point \( \mathbf{p}^{(W)}_i \)
  - Given the estimate robot pose \( \mathbf{x} \rightarrow \mathbf{T}^{(W)}_{WR} \)
  - i.e., \( \mathbf{p}^{(R)}_i \) in polar coordinates wrt

**Measurement with noise**

- \( \mathbf{p}^{(R)}_i = (\mathbf{T}^{(W)}_{WR})^{-1} \mathbf{p}^{(W)}_i \)
- \( \rho_i = \sqrt{p_{ix}^{(R)} + p_{iy}^{(R)}^2} \)
- \( \theta_i = \text{atan2}(p_{iy}^{(R)}, p_{ix}^{(R)}) \)
- \( h_i(\mathbf{x}, \mathbf{p}^{(W)}_i, \delta_i) = \begin{cases} 
\tilde{\rho}_i = \sqrt{\mathbf{p}_{ix}^{(R)} + \mathbf{p}_{iy}^{(R)}^2} + \delta_{\rho_i} \\
\tilde{\theta}_i = \text{atan2}(\mathbf{p}_{iy}^{(R)}, \mathbf{p}_{ix}^{(R)}) + \delta_{\theta_i} 
\end{cases} \)
- \( \delta_i = [\delta_{\rho_i}, \delta_{\theta_i}]^T \sim \mathcal{N}(0, \mathbf{Q}_i) \)
Measurement & Update Step - Jacobians

**Measurement Equation**

\[ h_i(x, p_i^{(W)}, \delta_i) = \left\{ \begin{array}{l} \tilde{\rho}_i = \sqrt{p_{ix}^{(R)} + p_{iy}^{(R)}} + \delta \rho_i \\ \tilde{\theta}_i = \text{atan2}(p_{iy}^{(R)}, p_{ix}^{(R)}) + \delta \theta_i \end{array} \right. \]

\[ p_i^{(R)} = (T_{WR})^{-1} p_i^{(W)} \]

**EKF Jacobians**

- **H_i** = \[ \frac{\partial h_i(x, p, \delta)}{\partial x} \bigg|_{x=\mu_{t-1}, p=p_i^{(W)}, \delta_i=0} \]

  derivate of the measurement function w.r.t. state variables

- **M_i** = \[ \frac{\partial h_i(x, p, \delta)}{\partial \delta} \bigg|_{x=\mu_{t-1}, p=p_i^{(W)}, \delta_i=0} \]

  derivate of the measurement function w.r.t. noise variables

**Jacobians**

\[ H_i = \begin{bmatrix} \frac{\partial h_1(x, p, \delta)}{\partial x} & \frac{\partial h_1(x, p, \delta)}{\partial y} & \frac{\partial h_1(x, p, \delta)}{\partial \theta} \\ \frac{\partial h_2(x, p, \delta)}{\partial x} & \frac{\partial h_2(x, p, \delta)}{\partial y} & \frac{\partial h_2(x, p, \delta)}{\partial \theta} \end{bmatrix} = \ldots \]

\[ M_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

**Some Notes**

- Not all \( h_i(\cdot) \) are valid
  - e.g., \( \rho_i \notin [\rho_{\text{min}}, \rho_{\text{max}}] \)
  - e.g., \( \theta_i \notin [\theta_{\text{min}}, \theta_{\text{max}}] \)
- We select a subset of \( h_i(\cdot) \)
**Measurement & Update Step - Measurement Details**

- **Cyan**: the *predicted measure*, $h_i(\cdot)$
- **Green**: the real map point in robot coordinates
- **Blue**: the noisy sensor measurement $z_i$

**Ellipses**: given by covariance

$$S_t = H_t \Sigma_t H_t^T + M_t Q_t M_t^T$$

**Innovation**: $z_i - h_i(\cdot)$
The measurements

- \( h_i(\cdot), z_i(\cdot), H_i, M_i(\cdot), Q_i(\cdot) \)
  feasible measurements and Jacobians
- How to update?

The complete measurements

\[
\begin{align*}
  h(x, m, \delta) &= \begin{cases} 
    h_1(x, p_1^{(W)}, \delta_1) \\
    h_2(x, p_2^{(W)}, \delta_2) \\
    \vdots \\
    h_m(x, p_m^{(W)}, \delta_m)
  \end{cases} \\
  \delta &= \begin{bmatrix} 
    \delta_1^T \\
    \delta_2^T \\
    \vdots \\
    \delta_m^T
  \end{bmatrix}^T
\end{align*}
\]

The update

\[
\begin{align*}
  h &= [h_1^T \ h_2^T \ \cdots \ h_m^T]^T \\
  H &= [H_1^T \ H_2^T \ \cdots \ H_m^T]^T \\
  z &= [z_1^T \ z_2^T \ \cdots \ z_m^T]^T \\
  M &= \begin{bmatrix} 
    M_1 & 0 & \cdots & 0 \\
    0 & M_2 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & \cdots & M_{m-1} & 0 \\
    0 & \cdots & 0 & M_m
  \end{bmatrix} \\
  Q &= \begin{bmatrix} 
    Q_1 & 0 & \cdots & 0 \\
    0 & Q_2 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & \cdots & Q_{m-1} & 0 \\
    0 & \cdots & 0 & Q_m
  \end{bmatrix}
\end{align*}
\]
Measurement & Update Step - Some Steps - 1

**IN THE WORLD**

Covariance on robot pose is reduced
Covariance on measurement is reduced due to the minor uncertainty in the pose.
THE COMPLETE PATH
The complete path - Worse Sensor

Estimated trajectory is less precise, covariance on robot pose is bigger
Correspondences

**Correspondences**

- Correspondences are known → this is uncommon in real environments
- If correspondences are unknown we have to perform the data association

**Data association**

- Given a set of measurements \( \{z_i\}, i = 1 : m \)
- Given a set of measurements prediction \( \{h_j\}, j = 1 : w \)
- We have to select correspondences \( c_{ij} \)

**“Dummy” approach**

1. \( k = 1 \)
2. Select \( w \) such that \( z_w \) closest to \( h_k \)
3. Remove \( z_w \) from \( \{z_i\} \)
4. Repeat from 2 and increment \( k \)

**Maximum Likelihood approach**

1. Select a sufficient number of landmarks in the map
2. The correspondence is chosen by maximizing the likelihood of the measurement \( z_t^i \) given any possible landmark \( k_i \) in the map

\[
k_t = \max_i p(z_t|c_{1:t}, m, z_{1:t-1}, u_{1:t}, \{k_t\}_i)
\]
**Given**
- Given $A$, $B$ coordinates
- Distance to $(x, y)$
- Suppose to know covariance $\Sigma$
- i.e., $\sim \mathcal{N}(\mu = [x, y], \Sigma)$

**Euclidean Distance**
- $B$ is closest to $x, y$
- $A$ is far

**Mahalanobis Distance**
- $D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$
- Squared distance weighted for the inverse of covariance
- $D^2(A) < D^2(B)$, $A$ is inside the covariance ellipse
- It is a scaled and rotated distance
- Same probability $= $ same distance
- $D^2$ is distributed as a $\chi^2(n)$
Data association

DATA ASSOCIATION WITH $D^2$

1. $k = 1$
2. Select $w$ such that $z_w$ closest to $h_k$ in $D^2(z_w, h_k)$
3. Remove $z_w$ from $\{z_i\}$
4. Repeat from 2 and increment $k$

Further, when minimum distance is too high, stop association algorithm

IS IT THE RIGHT APPROACH?

- Is better than the Euclidean distance data association
- Could performs wrong associations
- It consider only individual compatibility
- Resulting in a bad localization
- Better approach will consider joint compatibility or performs multi hypothesis
MHT represents belief by multiple Gaussian:

\[
bel(x_t) = \frac{1}{\sum_j \phi_{t,j}} \sum_j \phi_{t,j} \det(2\pi \Sigma_{t,j})^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (x_t - \mu_{j,t})^T \Sigma_{j,t}^{-1} (x_t - \mu_{j,t})\right)
\]

where \(\phi_{t,j} > 0\)

- Each component is tracked by a Kalman filter
- The number of mixture components grows exponentially over time
- Pruning: every component whose relative weight \(\frac{\phi_{t,l}}{\sum_m \phi_{t,m}}\) is smaller than a threshold is removed
- Not easy to be implemented
- Better handled by particle filter
Outline

1. Kalman Filter
2. K.F. Example
3. E.K.F.
4. EKF Loc. - Prediction
5. EKF Loc. - Update
6. Correspondences
7. U.K.F.
8. UKF Loc.
9. Considerations
Unscented Kalman Filter

**Kalman Filter Main Issue**

- Taylor expansion is the simplest form of linearization
- Other approaches have found to be superior
- *moments matching*
  - linearization preserves the true mean and covariance of the posterior
  - Lead to the assumed density filter (ADF)
- Unscented Kalman Filter (UKF)
  - stochastic linearization through the use of weighted statistical linear regression process
Linearization – Unscented Kalman Filter

**Given** \( x \sim \mathcal{N}(\mu, \Sigma) \)

- Do not use Taylor expansion
- Identify \( 2n + 1 \) *sigma* points from the Gaussian

\[
\chi^{[0]} = \mu \\
\chi^{[i]} = \mu \pm \left( \sqrt{(n + \lambda)\Sigma} \right)_i, \ \forall i = 1, \ldots, n
\]

where \( \lambda \) is a scaling parameter:

- \( \lambda = \alpha^2(n + k) - n \)
- \( 0 \leq \alpha \leq 1 \)
- \( k \) scaling factor, i.e., extension in the initial distribution (may be 0)

- Two weights are associated to each sigma point

\[
w^{[0]}_m = \frac{\lambda}{n + \lambda} \quad \text{mean weight}
\]
\[
w^{[0]}_c = \frac{\lambda}{n + \lambda} + \left( 1 - \alpha^2 + \beta \right) \quad \text{covariance weight}
\]
\[
w^{[i]}_m = w^{[i]}_c = \frac{1}{2(n + \lambda)}, \ \forall i = 1, \ldots, 2n
\]

\( \beta \), a parameter that depends on a priori knowledge on the distribution

(for Gaussians, one can show that \( \beta = 2 \) is optimal)
Pass the sigma points through the nonlinear function $g(\cdot)$, in order to verify how $g$ changes the shape of the Gaussian

$$y^{[i]} = g\left(\chi^{[i]}\right)$$

Recover mean and covariance

$$\mu' = \sum_{i=0}^{2n} w_m y^{[i]}$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} \left(y^{[i]} - \mu'\right) \left(y^{[i]} - \mu'\right)^T$$

$$y \sim \mathcal{N}(\cdots)$$
EKF to UKF

**EKF**
(non–additive noise)

**STATE TRANSITION**
\[ \mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{\epsilon}_t) \]

**MEASUREMENT PROBABILITY**
\[ \mathbf{z}_t = h(\mathbf{x}_t, \delta_t) \]

**PREDICTION STEP**
\[ \overline{\mu}_t = g(\mu_{t-1}, \mathbf{u}_t, 0) \]
\[ \overline{\Sigma}_t = G_t \Sigma_{t-1} G^T_t + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T \]

**UPDATE STEP**
\[ \mathbf{K}_t = \overline{\Sigma}_t H^T_t (H_t \overline{\Sigma}_t H^T_t + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T)^{-1} \]
\[ \mu_t = \overline{\mu}_t + \mathbf{K}_t (\mathbf{z}_t - h(\overline{\mu}_t, 0)) \]
\[ \Sigma_t = (I - \mathbf{K}_t H_t) \overline{\Sigma}_t \]

**UKF**
(additive noise)

**STATE TRANSITION**
\[ \mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{\epsilon}_t \]

**MEASUREMENT PROBABILITY**
\[ \mathbf{z}_t = h(\mathbf{x}_t) + \delta_t \]

**PREDICTION STEP**
\[ \overline{\mu}_t = \sum_{i=0}^{2n} w_m [\chi^*]_i \mathbf{\chi}_t \]
\[ \overline{\Sigma}_t = \sum_{i=0}^{2n} w_c [\chi^*]_i (\chi^* - \overline{\mu}_t) (\chi^* - \overline{\mu}_t)^T + \mathbf{R}_t \]

**UPDATE STEP**
\[ \mathbf{K}_t = \overline{\Sigma}^{x,z} S_t^{-1} \]
\[ \mu_t = \overline{\mu}_t - \mathbf{K}_t (\mathbf{z}_t - \hat{\mathbf{z}}_t) \]
\[ \Sigma_t = \overline{\sigma}_t - \mathbf{K}_t S_t \mathbf{K}_t^T \]
UKF – The algorithm

UNSCENTED KALMAN FILTER

Algorithm Unscented_Kalman_filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

\[
\begin{align*}
X_{t-1} &= (μ_{t-1}, μ_{t-1} + γ\sqrt{Σ_{t-1}}, μ_{t-1} - γ\sqrt{Σ_{t-1}}) \\
X_t^* &= g(u_t, X_{t-1}) \\
μ_t &= \sum_{i=0}^{2n} w_m^{[i]}X_t^{*[i]} \\
Σ_t &= \sum_{i=0}^{2n} w_c^{[i]}(X_t^{*[i]} - μ_t)(X_t^{*[i]} - μ_t)^T + R_t \\
\bar{X}_t &= (μ_t, μ_t + γ\sqrt{Σ_t}, μ_t - γ\sqrt{Σ_t}) \\
\bar{Z}_t &= h(\bar{X}_t) \\
\bar{z}_t &= \sum_{i=0}^{2n} w_m^{[i]}\bar{Z}_t^{[i]} \\
S_t &= \sum_{i=0}^{2n} w_c^{[i]}(\bar{Z}_t^{[i]} - \bar{z}_t)(\bar{Z}_t^{[i]} - \bar{z}_t)^T + Q_t \\
Σ_{x,z} &= \sum_{i=0}^{2n} w_c^{[i]}(X_t^{[i]} - μ_t)(\bar{Z}_t^{[i]} - \bar{z}_t)^T \\
K_t &= Σ_{x,z} S_t^{-1} \\
μ_t &= \bar{μ}_t + K_t(\bar{z}_t - \bar{z}_t) \\
Σ_t &= Σ_t - K_t S_t K_t^T \\
\text{return } μ_t, Σ_t
\end{align*}
\]
The maintenance of a good Gaussian approximation depends on the variance of $x$. 
The maintenance of a good Gaussian approximation depends on the variance of $x$. 

$y \sim \mathcal{N}(\cdots)$
Linearization via Unscented Kalman Filter

Dependency of approximation quality on uncertainty

The maintenance of a good Gaussian approximation depends on the linearization point
The maintenance of a good Gaussian approximation depends on the linearization point.
UKF - Summary

**Complexity**

- Highly Efficient
  - Same complexity of EKF
  - In practical applications is slower (with a constant factor)

- Better linearization than EKF
  - Accurate in the first two term of Taylor expansion
  - EKF only first term

- Derivative–free
  - No Jacobians

- Still not optimal. May diverge if nonlinearities are large.
Localization with U.K.F.

**Localization**

- Special case of Markov Localization
- UKF can treat the *pose tracking* problem (like EKF and MHT)
- We can consider *uncertainty* (or *belief*) locally Gaussian
- Derivative–free
- Linearization is performed through unscented transform
- Landmarks are uniquely identifiable

- Algorithm is reported in the book [Probabilistic Robotics, 2005]
E.K.F. v.s. U.K.F.

**State transition - Prediction**
- The next robot position using motion information

**Measurement - Update**
- Sense of a map landmark
  \[ z_t = \{ r_t, \phi_t, s_t \} \]
  i.e., sense relative distance and bearing from the marker

**Action**
- The next robot position
  \[ u_t = (v_t, w_t) \]

**Map and correspondences**
- Six uniquely colored markers around the field
- Landmarks are uniquely identifiable e.g., different colors
- Identify an unique marker at each time \( t \)
E.K.F. v.s. U.K.F. (trajectory)

- Robot’s trajectory according to motion model (dashed line)
- The resulting trajectory (solid line in Figure a)
- EKF (b) and UKF (c) have close behavior
- UKF performs slightly better
E.K.F. v.s. U.K.F. (Linearization effects)

- EKF misleads both in the localization of the mean and in the shape of the covariance
- However
  - The mean predicted by EKF is always in the location predicted by the control
  - The mean predicted by UKF varies since it is extracted from the sigma points
In EKF/UKF with unknown associations and ML, the cycle over all landmarks may be heavy. In practice, one has to limit the number of tests to be done, some observations being very far from the predicted ones. Combine with adequate data structure for spatial memory (e.g., quadtrees). We require data association techniques to ensure that mutual exclusion is verified, 
\[ i = j \rightarrow k(i) = k(j) \]
Filtering outliers is important in the case of unknown correspondences; a simple way to do it is to follow the ML.
Considerations

**EKF/UKF**

- UKF and EKF are more adapted to the *pose tracking* problem,
- the versions of these filters without knowledge of the correspondences require small uncertainty volumes to work, otherwise association errors are more likely, with catastrophic consequences,
- linearization or UT require *limited uncertainty* volumes.

**MHT**

The MHT a priori can handle the *global localization* problem

- initialize the mixture with the first observations and their possible correspondences,
- theoretically, if one also incorporate explicitly the kidnapping hypothesis, it could be possible to handle the kidnapping problem,
- it is more *robust* by definition to correspondence errors,
- the processing of the each component of the mixture corresponds to an EKF or an UKF.
Considerations

- General rule
  - more features make the uncertainty volumes smaller,
  - it is the reason why EKF or UKF work well,
  - but the denser in features is the world, the more likely are correspondences errors!

**Negative Information**

- In none of these algorithms, we use negative information, i.e., the absence of a feature when one expects to see it (relevant information),
- In EKF and UKF may be difficult to exploit
- in the MHT, one could penalize the components of the mixture through their weights when some no-observations occur.
Reference

“Probabilistic Robotics”

(Intelligent Robotics and Autonomous Agents series)

The MIT Press (2005)

Chapters 3, 7.