Goal: Simultaneous Localization and Mapping (SLAM)

- Building a map and locating the robot in the map at the same time
- Chicken-or-egg problem

SLAM is a State Estimation Problem

- Estimate the map and robot’s pose
- Bayes filter is one tool for state estimation

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

**Correction**
\[
bel(x_t) = \eta \, p(z_t \mid x_t) \, \overline{bel}(x_{t-1})
\]
Gaussians

- Everything is Gaussian

\[ p(x) = \det(2\pi \Sigma)^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]

Linear Model

- The Kalman filter assumes a linear transition and observation model
- Zero mean Gaussian noise

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

\[ z_t = C_t x_t + \delta_t \]

Properties: Marginalization and Conditioning

- Given

\[ x = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \quad p(x) = \mathcal{N} \]

- The marginals are Gaussians

\[ p(x_a) = \mathcal{N} \quad p(x_b) = \mathcal{N} \]

- as well as the conditionals

\[ p(x_a \mid x_b) = \mathcal{N} \quad p(x_b \mid x_a) = \mathcal{N} \]

Components of a Kalman Filter

- Matrix \((n \times n)\) that describes how the state evolves from \(t-1\) to \(t\) without controls or noise.
- Matrix \((n \times l)\) that describes how the control \(u_t\) changes the state from \(t-1\) to \(t\).
- Matrix \((k \times n)\) that describes how to map the state \(x_t\) to an observation \(z_t\).
- 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.
**Linear Motion Model**

- Motion under Gaussian noise leads to

\[
p(x_t | u_t, x_{t-1}) = \?
\]

**Linear Observation Model**

- Measuring under Gaussian noise leads to

\[
p(z_t | x_t) = \?
\]

**Linear Motion Model**

- Motion under Gaussian noise leads to

\[
p(x_t | u_t, x_{t-1}) = \det(2\pi R_t)^{-\frac{1}{2}} e^{\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)}
\]

- \( R_t \) describes the noise of the motion

**Linear Observation Model**

- Measuring under Gaussian noise leads to

\[
p(z_t | x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} e^{\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t)}
\]

- \( Q_t \) describes the measurement noise
Everything stays Gaussian

- Given an initial Gaussian belief, the belief is always Gaussian

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

\[ bel(x_t) = \eta \, p(z_t \mid x_t) \, \overline{bel}(x_{t-1}) \]

- Proof is non-trivial
  (see Probabilistic Robotics, Sec. 3.2.4)

Kalman Filter Algorithm

1: \text{Kalman filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t):
2: \quad \bar{\mu}_t = A_t \mu_{t-1} + B_t \, u_t
3: \quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
4: \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}
5: \quad \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)
6: \quad \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t
7: \quad \text{return } \mu_t, \Sigma_t

1D Kalman Filter Example (1)

- prediction
- measurement
- correction

It's a weighted mean!

1D Kalman Filter Example (2)

- prediction
- measurement
- correction
Kalman Filter Assumptions

- Gaussian distributions and noise
- Linear motion and observation model

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

What if this is not the case?

Non-linear Dynamic Systems

- Most realistic problems (in robotics) involve nonlinear functions

\[ x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \]
\[ z_t = C_t x_t + \delta_t \]
\[ x_t = g(u_t, x_{t-1}) + \epsilon_t \]
\[ z_t = h(x_t) + \delta_t \]

Linearity Assumption Revisited

Non-Linear Function

Non-Gaussian!
Non-Gaussian Distributions

- The non-linear functions lead to non-Gaussian distributions
- Kalman filter is not applicable anymore!

What can be done to resolve this?

EKF Linearization: First Order Taylor Expansion

- Prediction:
  \[ g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]
  \[ =: G_t \]

- Correction:
  \[ h(x_t) \approx h(\mu_t) + \frac{\partial h(\mu_t)}{\partial x_t} (x_t - \mu_t) \]
  \[ =: H_t \] Jacobian matrices

Reminder: Jacobian Matrix

- It is a non-square matrix \( n \times m \) in general
- Given a vector-valued function
  \[ g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix} \]
- The Jacobian matrix is defined as
  \[ G_x = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_m} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_m} \end{pmatrix} \]
**Reminder: Jacobian Matrix**
- It is the orientation of the tangent plane to the vector-valued function at a given point.
- Generalizes the gradient of a scalar valued function.

**EKF Linearization: First Order Taylor Expansion**
- **Prediction:**
  \[ g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]
  \[ =: G_t \]
- **Correction:**
  \[ h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t) \]
  \[ =: H_t \] (Linear functions!)

**Linearity Assumption Revisited**

**Non-Linear Function**
The linearized model leads to

\[ p(x_t \mid u_t, x_{t-1}) \approx \det(2\pi R_t)^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (x_t - g(u_t, \mu_{t-1}) - G_t (x_{t-1} - \mu_{t-1}))^T R_t^{-1} (x_t - g(u_t, \mu_{t-1}) - G_t (x_{t-1} - \mu_{t-1})) \right) \]

- \( R_t \) describes the noise of the motion
**Linearized Observation Model**

- The linearized model leads to
  \[
  p(z_t \mid x_t) = \det (2\pi Q_t)^{-\frac{1}{2}} \\
  \exp \left( -\frac{1}{2} (z_t - h(\mu_t) - H_t (x_t - \mu_t))^T Q_t^{-1} (z_t - h(\mu_t) - H_t (x_t - \mu_t)) \right)
  \]

- \(Q_t\) describes the measurement noise

**Extended Kalman Filter Algorithm**

1. \(
\text{Extended} \_\text{Kalman} \_\text{filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\
\)
2. \(\bar{\mu}_t = g(u_t, \mu_{t-1})\)
3. \(\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t\)
4. \(K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}\)
5. \(\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))\)
6. \(\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t\)
7. return \(\mu_t, \Sigma_t\)

**Extended Kalman Filter Summary**

- Extension of the Kalman filter
- Ad-hoc solution to handle the non-linearities
- Performs local linearizations
- Works well in practice for moderate non-linearities
- Complexity: \(O(k^{2.4} + n^2)\)

**Literature**

**Kalman Filter and EKF**

- Thrun et al.: “Probabilistic Robotics”, Chapter 3
- Schön and Lindsten: “Manipulating the Multivariate Gaussian Density”
- Welch and Bishop: “Kalman Filter Tutorial”