Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

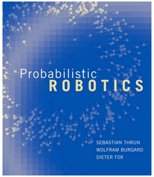
Institute for AI and Autonomous Systems

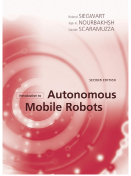
Kalman Filter-Based Localisation

Localising with a Gaussian Uncertainty Model

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure



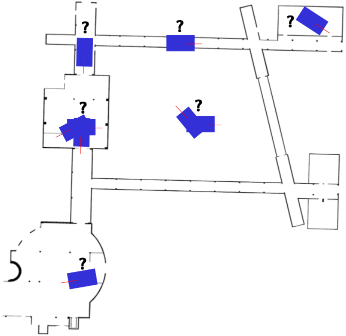


- Localisation preliminaries
- Kalman filter

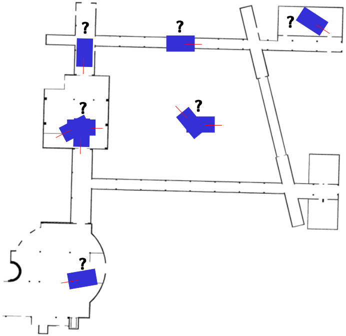
Localisation Preliminaries

What is Localisation?

- In order for a robot to move purposefully (e.g. find path plans as discussed in the last lecture), it needs to **know its whereabouts in the environment**

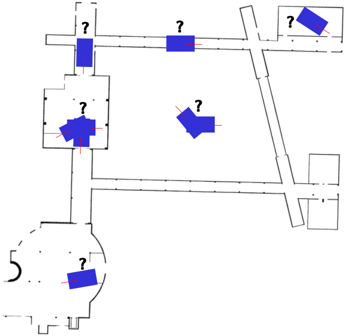


What is Localisation?



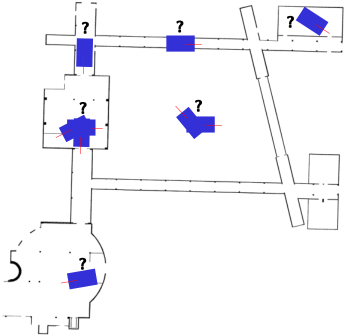
- In order for a robot to move purposefully (e.g. find path plans as discussed in the last lecture), it needs to **know its whereabouts in the environment**
- The process that determines a robot's pose in an environment is called **localisation**

What is Localisation?



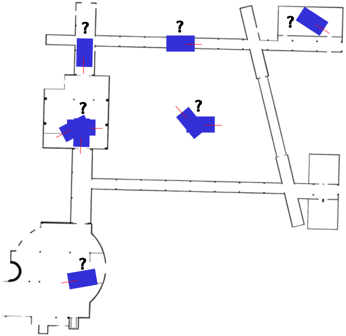
- In order for a robot to move purposefully (e.g. find path plans as discussed in the last lecture), it needs to **know its whereabouts in the environment**
- The process that determines a robot's pose in an environment is called **localisation**
- The localisation process is performed **with respect to a given map** — thus, **localisation requires a map to be given**
  - The process of creating a map and localising is called **simultaneous localisation and mapping (SLAM)**

What is Localisation?



- In order for a robot to move purposefully (e.g. find path plans as discussed in the last lecture), it needs to **know its whereabouts in the environment**
- The process that determines a robot's pose in an environment is called **localisation**
- The localisation process is performed **with respect to a given map** — thus, **localisation requires a map to be given**
  - The process of creating a map and localising is called **simultaneous localisation and mapping (SLAM)**
- Since a robot moves around, the localisation estimate needs to be continuously updated based on the **known motion commands** and **any relevant environment observations**

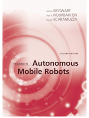
What is Localisation?

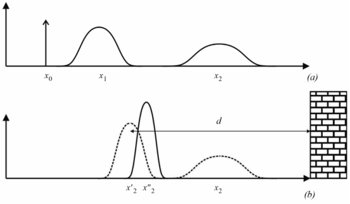


- In order for a robot to move purposefully (e.g. find path plans as discussed in the last lecture), it needs to **know its whereabouts in the environment**
- The process that determines a robot's pose in an environment is called **localisation**
- The localisation process is performed **with respect to a given map** — thus, **localisation requires a map to be given**
  - The process of creating a map and localising is called **simultaneous localisation and mapping (SLAM)**
- Since a robot moves around, the localisation estimate needs to be continuously updated based on the **known motion commands** and **any relevant environment observations**

Localisation is a process of estimating the pose of a mobile robot in a given environment as the robot moves around and collects sensor measurements

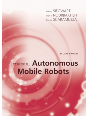
Pose Uncertainty and Belief

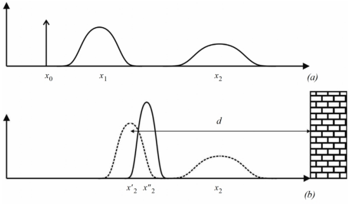




- Due to noisy motions and sensor measurements, **a robot is typically never fully certain about its pose in the environment**
  - The pose estimate is always associated with some **uncertainty**

Pose Uncertainty and Belief



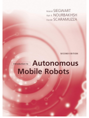


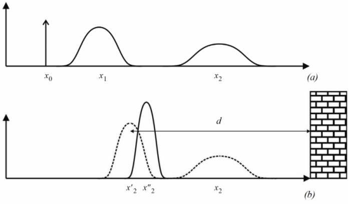
- Due to noisy motions and sensor measurements, **a robot is typically never fully certain about its pose in the environment**
  - The pose estimate is always associated with some **uncertainty**
- For this reason, pose estimates are represented by a **belief** $\text{bel}(\boldsymbol{x}_t)$, which models the probability distribution

$$\text{bel}(\boldsymbol{x}_t) = \eta \, p(\boldsymbol{x}_t | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}, \boldsymbol{x}_{0:t-1})$$

where $\boldsymbol{x}$ represents a state, $\boldsymbol{u}$ is a control signal, $\boldsymbol{z}$ is a measurement, and η is a normalisation constant

Pose Uncertainty and Belief





- Due to noisy motions and sensor measurements, **a robot is typically never fully certain about its pose in the environment**
  - The pose estimate is always associated with some **uncertainty**
- For this reason, pose estimates are represented by a **belief** $\text{bel}\left(\boldsymbol{x}_t\right)$, which models the probability distribution

$$\text{bel}\left(\boldsymbol{x}_t\right) = \eta\, p\left(\boldsymbol{x}_t | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}, \boldsymbol{x}_{0:t-1}\right)$$

  where $\boldsymbol{x}$ represents a state, $\boldsymbol{u}$ is a control signal, $\boldsymbol{z}$ is a measurement, and η is a normalisation constant
- The belief is **continuously updated** as a robot moves around and collects measurements of the environment

Kidnapped Robot Problem

- In the context of localisation, it is important to consider **whether a robot can recover from catastrophic localisation failures**

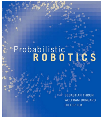
Kidnapped Robot Problem

- In the context of localisation, it is important to consider **whether a robot can recover from catastrophic localisation failures**
- The kidnapped robot problem refers to **a complete localisation loss** — as if a robot has been kidnapped by someone and brought to a completely different location than where it started

Kidnapped Robot Problem

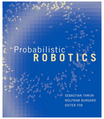
- In the context of localisation, it is important to consider **whether a robot can recover from catastrophic localisation failures**
- The kidnapped robot problem refers to **a complete localisation loss** — as if a robot has been kidnapped by someone and brought to a completely different location than where it started
- Ideally, **a localisation method should be robust to the kidnapped robot problem**
  - Robustness requires a possibility to discard the current localisation estimate as wrong so that a new estimate can be made

Recursive State Estimation Using a Bayes Filter



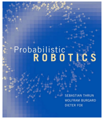
- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**

Recursive State Estimation Using a Bayes Filter



- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**
- The Bayes filter continuously performs two steps:

Recursive State Estimation Using a Bayes Filter



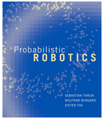
- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**
- The Bayes filter continuously performs two steps:

Control update (prediction)

The belief is updated based on a performed motion $\boldsymbol{u}_t$:

$$\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \int p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) \text{bel}\left(\boldsymbol{x}_{t-1}\right) d\boldsymbol{x}_{t-1}$$

Recursive State Estimation Using a Bayes Filter



- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**

- The Bayes filter continuously performs two steps:

Control update (prediction)

The belief is updated based on a performed motion $\boldsymbol{u}_t$:

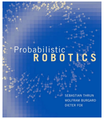
$$\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \int p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) \text{bel}\left(\boldsymbol{x}_{t-1}\right) d\boldsymbol{x}_{t-1}$$

Measurement update (correction)

The belief is corrected based on a sensor measurement $\boldsymbol{z}_t$:

$$\text{bel}\left(\boldsymbol{x}_t\right) = \eta\, p\left(\boldsymbol{z}_t | \boldsymbol{x}_t\right) \overline{\text{bel}}\left(\boldsymbol{x}_t\right)$$

Recursive State Estimation Using a Bayes Filter



- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**
- The Bayes filter continuously performs two steps:

Control update (prediction)

The belief is updated based on a performed motion $\boldsymbol{u}_t$:

$$\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \int p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) \text{bel}\left(\boldsymbol{x}_{t-1}\right) d\boldsymbol{x}_{t-1}$$

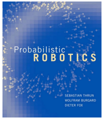
Measurement update (correction)

The belief is corrected based on a sensor measurement $\boldsymbol{z}_t$:

$$\text{bel}\left(\boldsymbol{x}_t\right) = \eta\, p\left(\boldsymbol{z}_t | \boldsymbol{x}_t\right) \overline{\text{bel}}\left(\boldsymbol{x}_t\right)$$

- The control update **increases the pose uncertainty**, while the measurement update **decreases it**

Recursive State Estimation Using a Bayes Filter



- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**

- The Bayes filter continuously performs two steps:

Control update (prediction)

The belief is updated based on a performed motion $\boldsymbol{u}_t$:

$$\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \int p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) \text{bel}\left(\boldsymbol{x}_{t-1}\right) d\boldsymbol{x}_{t-1}$$

Measurement update (correction)

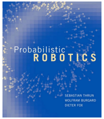
The belief is corrected based on a sensor measurement $\boldsymbol{z}_t$:

$$\text{bel}\left(\boldsymbol{x}_t\right) = \eta\, p\left(\boldsymbol{z}_t | \boldsymbol{x}_t\right) \overline{\text{bel}}\left(\boldsymbol{x}_t\right)$$

- The control update **increases the pose uncertainty**, while the measurement update **decreases it**

- Both update equations are based on the **Markov assumption**, according to which:
  - $\boldsymbol{x}_t$ only depends on $\boldsymbol{x}_{t-1}$ and $\boldsymbol{u}_t$
  - $\boldsymbol{z}_t$ only depends on $\boldsymbol{x}_t$

Recursive State Estimation Using a Bayes Filter



- The process of continuously updating the belief based on performed motions and recorded measurements is called **recursive state estimation**; this is performed using a **Bayes filter**
- The Bayes filter continuously performs two steps:

Control update (prediction)

The belief is updated based on a performed motion $\boldsymbol{u}_t$:

$$\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \int p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) \text{bel}\left(\boldsymbol{x}_{t-1}\right) d\boldsymbol{x}_{t-1}$$

Measurement update (correction)

The belief is corrected based on a sensor measurement $\boldsymbol{z}_t$:

$$\text{bel}\left(\boldsymbol{x}_t\right) = \eta \, p\left(\boldsymbol{z}_t | \boldsymbol{x}_t\right) \overline{\text{bel}}\left(\boldsymbol{x}_t\right)$$

- The control update **increases the pose uncertainty**, while the measurement update **decreases it**
- Both update equations are based on the **Markov assumption**, according to which:
  - $\boldsymbol{x}_t$ only depends on $\boldsymbol{x}_{t-1}$ and $\boldsymbol{u}_t$
  - $\boldsymbol{z}_t$ only depends on $\boldsymbol{x}_t$
- More background details behind the Bayes filter are covered in "Mathematics for Robotics and Control"

Recursive Bayes Filter Family

- The Bayes filter is not a single algorithm, but **a family of algorithms**, each of which is based on different assumptions

Recursive Bayes Filter Family

- The Bayes filter is not a single algorithm, but **a family of algorithms**, each of which is based on different assumptions
- We will particularly distinguish between the following variants of the recursive Bayes filter as applied to the robot localisation problem:
  - **Discrete Bayes filter**: Applicable when **the state is discretised** (discussed in MRC)

Recursive Bayes Filter Family

- The Bayes filter is not a single algorithm, but **a family of algorithms**, each of which is based on different assumptions
- We will particularly distinguish between the following variants of the recursive Bayes filter as applied to the robot localisation problem:
  - **Discrete Bayes filter**: Applicable when **the state is discretised** (discussed in MRC)
  - **Kalman filter**: Assumes that **the state is governed by a Gaussian distribution** (discussed today)

Recursive Bayes Filter Family

- The Bayes filter is not a single algorithm, but **a family of algorithms**, each of which is based on different assumptions
- We will particularly distinguish between the following variants of the recursive Bayes filter as applied to the robot localisation problem:
  - **Discrete Bayes filter**: Applicable when **the state is discretised** (discussed in MRC)
  - **Kalman filter**: Assumes that **the state is governed by a Gaussian distribution** (discussed today)
  - **Particle filter**: Does not make an assumption about the underlying state distribution; **represents multiple hypotheses about the state by a set of particles** (discussed next time)

Kalman Filter

Kalman Filter Preliminaries

- The Kalman filter is a Bayes filter that can be used for **continuous state and measurement spaces**

Kalman Filter Preliminaries

- The Kalman filter is a Bayes filter that can be used for **continuous state and measurement spaces**
- In a Kalman filter, the belief about the state is represented by a **Gaussian distribution** $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$
  - The estimation thus performs updates of the mean $\boldsymbol{\mu}$ and the covariance matrix Σ

Kalman Filter Preliminaries

- The Kalman filter is a Bayes filter that can be used for **continuous state and measurement spaces**
- In a Kalman filter, the belief about the state is represented by a **Gaussian distribution** $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$
  - The estimation thus performs updates of the mean $\boldsymbol{\mu}$ and the covariance matrix Σ
- Recall that a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ for an n-dimensional $\boldsymbol{x}$ has the form

$$\mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}$$

Kalman Filter Preliminaries

- The Kalman filter is a Bayes filter that can be used for **continuous state and measurement spaces**
- In a Kalman filter, the belief about the state is represented by a **Gaussian distribution** $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$
  - The estimation thus performs updates of the mean $\boldsymbol{\mu}$ and the covariance matrix Σ
- Recall that a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ for an n-dimensional $\boldsymbol{x}$ has the form

$$\mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}$$

- Considering a state $\boldsymbol{x}_t$ and an estimate $\hat{\boldsymbol{x}}_t$, the filter is **optimal** in the sense that it minimises the mean squared error

$$MSE(\boldsymbol{x}_t, \hat{\boldsymbol{x}}) = (\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)^T (\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)$$

Linear Motion and Measurement Models

- The Kalman filter assumes that **the motion and measurement models are both linear**, and that **the initial state distribution is governed by a Gaussian distribution**; under these assumptions, **the posterior state estimate is also a Gaussian**

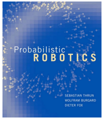
Linear Motion and Measurement Models

- The Kalman filter assumes that **the motion and measurement models are both linear**, and that **the initial state distribution is governed by a Gaussian distribution**; under these assumptions, **the posterior state estimate is also a Gaussian**
- This means that **the motion model is governed by a linear system** of the form

$$\boldsymbol{x}_t = A_t \boldsymbol{x}_{t-1} + B_t \boldsymbol{u}_t + \boldsymbol{\epsilon}_t$$

where A_t is a state transition matrix, B_t is a process matrix, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$ is process noise

Linear Motion and Measurement Models



- The Kalman filter assumes that **the motion and measurement models are both linear**, and that **the initial state distribution is governed by a Gaussian distribution**; under these assumptions, **the posterior state estimate is also a Gaussian**

- This means that **the motion model is governed by a linear system** of the form

$$\boldsymbol{x}_t = A_t \boldsymbol{x}_{t-1} + B_t \boldsymbol{u}_t + \boldsymbol{\epsilon}_t$$

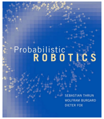
  where A_t is a state transition matrix, B_t is a process matrix, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$ is process noise

- The **measurement model is also given by a linear system**, which is represented as

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \boldsymbol{\delta}_t$$

  where C_t is a measurement matrix and $\boldsymbol{\delta}_t \sim \mathcal{N}(0, Q_t)$ is measurement noise

Kalman Filter Intuition



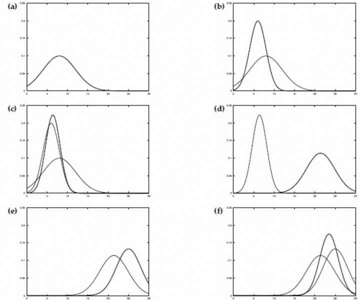
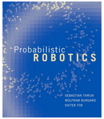


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

- A Kalman filter is a Bayes filter, so, at its core, it performs state and measurements updates

Kalman Filter Intuition



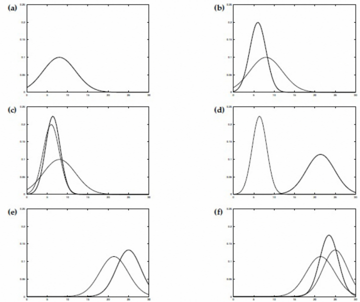
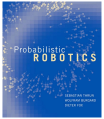


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

- A Kalman filter is a Bayes filter, so, at its core, it performs state and measurements updates
- Often, x_0 **is assumed to be known without ambiguity**; thus, it is represented by Dirac delta function, in which case $\boldsymbol{\mu}_0 = \delta\left(\boldsymbol{x}_0\right)$ and $\Sigma = \mathbf{0}$

Kalman Filter Intuition



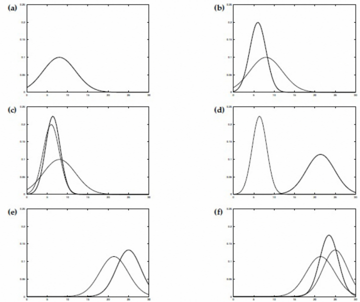
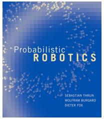


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

- A Kalman filter is a Bayes filter, so, at its core, it performs state and measurements updates
- Often, x_0 **is assumed to be known without ambiguity**; thus, it is represented by Dirac delta function, in which case $\boldsymbol{\mu}_0 = \delta(\boldsymbol{x}_0)$ and $\Sigma = \mathbf{0}$
- At any other time t, the state estimate is given by a Gaussian $\mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1})$, such that the objective is to **find an updated Gaussian** $\mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$
  - The estimate should incorporate a motion update and a measurement update, which are governed by the previously discussed linear motion model

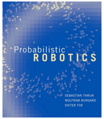
Kalman Filter Summary



1. **The state is updated** based on the controlled motion:

$$\overline{\boldsymbol{\mu}}_t = A_t \boldsymbol{\mu}_{t-1} + B_t \boldsymbol{u}_t$$

Kalman Filter Summary



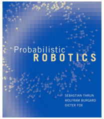
1. **The state is updated** based on the controlled motion:

$$\boxed{\overline{\boldsymbol{\mu}}_t} = A_t \boldsymbol{\mu}_{t-1} + B_t \boldsymbol{u}_t$$

2. **The covariance is updated** based on the controlled motion by considering the motion noise:

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

Kalman Filter Summary



1. **The state is updated** based on the controlled motion:

$$\overline{\boldsymbol{\mu}}_t = A_t \boldsymbol{\mu}_{t-1} + B_t \boldsymbol{u}_t$$

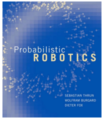
2. **The covariance is updated** based on the controlled motion by considering the motion noise:

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

3. **The Kalman gain is computed** using the updated covariance and the measurement noise:

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

Kalman Filter Summary



1. **The state is updated** based on the controlled motion:

$$\bar{\boldsymbol{\mu}}_t = A_t \boldsymbol{\mu}_{t-1} + B_t \boldsymbol{u}_t$$

2. **The covariance is updated** based on the controlled motion by considering the motion noise:

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

3. **The Kalman gain is computed** using the updated covariance and the measurement noise:

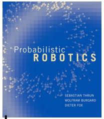
$$K_t = \bar{\Sigma}_t C_t^T \left(C_t \bar{\Sigma}_t C_t^T + Q_t \right)^{-1}$$

4. **The state and the covariance are updated** based on the measurement:

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t \left(\boldsymbol{z}_t - C_t \bar{\boldsymbol{\mu}}_t \right)$$

$$\Sigma_t = \left(I - K_t C_t \right) \bar{\Sigma}_t$$

Kalman Filter Derivation Sketch: Control Update



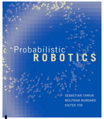
The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

Kalman Filter Derivation Sketch: Control Update

The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}(\boldsymbol{x}_t)$ by substituting for $p(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ and $\text{bel}(\boldsymbol{x}_{t-1})$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}(\boldsymbol{x}_t) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$

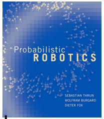
Kalman Filter Derivation Sketch: Control Update



The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ by substituting for $p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right)$ and $\text{bel}\left(\boldsymbol{x}_{t-1}\right)$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$
2. **Rewriting L_t in the form $L_t = L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) + L_t\left(\boldsymbol{x}_t\right)$** so that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta\, e^{-L_t\left(\boldsymbol{x}_t\right)} \int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$

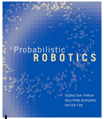
Kalman Filter Derivation Sketch: Control Update



The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ by substituting for $p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right)$ and $\text{bel}\left(\boldsymbol{x}_{t-1}\right)$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$
2. **Rewriting L_t in the form $L_t = L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) + L_t\left(\boldsymbol{x}_t\right)$** so that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \, e^{-L_t\left(\boldsymbol{x}_t\right)} \int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$
3. **Selecting $L_t\left(\boldsymbol{x}_t\right)$ to be a quadratic function with a constant $\int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$**; as a result, $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \, e^{-L_t\left(\boldsymbol{x}_t\right)}$

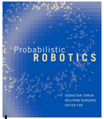
Kalman Filter Derivation Sketch: Control Update



The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ by substituting for $p\left(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right)$ and $\text{bel}\left(\boldsymbol{x}_{t-1}\right)$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$
2. **Rewriting L_t in the form $L_t = L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) + L_t\left(\boldsymbol{x}_t\right)$** so that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \, e^{-L_t\left(\boldsymbol{x}_t\right)} \int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$
3. **Selecting $L_t\left(\boldsymbol{x}_t\right)$ to be a quadratic function with a constant $\int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$**; as a result, $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \, e^{-L_t\left(\boldsymbol{x}_t\right)}$
4. **Determining $L_t\left(\boldsymbol{x}_t\right)$ from L_t and $L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right)$**; this results in a quadratic function, which means that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ is a Gaussian distribution

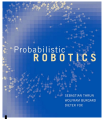
Kalman Filter Derivation Sketch: Control Update



The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}(\boldsymbol{x}_t)$ by substituting for $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ and $\text{bel}(\boldsymbol{x}_{t-1})$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}(\boldsymbol{x}_t) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$
2. **Rewriting L_t in the form $L_t = L_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) + L_t(\boldsymbol{x}_t)$** so that $\overline{\text{bel}}(\boldsymbol{x}_t) = \eta\, e^{-L_t(\boldsymbol{x}_t)} \int L_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)\, d\boldsymbol{x}_{t-1}$
3. **Selecting $L_t(\boldsymbol{x}_t)$ to be a quadratic function with a constant $\int L_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)\, d\boldsymbol{x}_{t-1}$**; as a result, $\overline{\text{bel}}(\boldsymbol{x}_t) = \eta\, e^{-L_t(\boldsymbol{x}_t)}$
4. **Determining $L_t(\boldsymbol{x}_t)$ from L_t and $L_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$**; this results in a quadratic function, which means that $\overline{\text{bel}}(\boldsymbol{x}_t)$ is a Gaussian distribution
   - The mean of the updated distribution is determined by the **minimum of $L_t(\boldsymbol{x}_t)$ over $\boldsymbol{x}_t$**

Kalman Filter Derivation Sketch: Control Update



The derivation of the Kalman filter equations is a bit involved, but can be summarised through the following steps:

1. **Expanding $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ by substituting for $p\left(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right)$ and $\text{bel}\left(\boldsymbol{x}_{t-1}\right)$**, both of which are Gaussian distributions; this results in a belief representation of the form $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta \int e^{-L_t} d\boldsymbol{x}_{t-1}$
2. **Rewriting L_t in the form $L_t = L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) + L_t\left(\boldsymbol{x}_t\right)$** so that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta\, e^{-L_t(\boldsymbol{x}_t)} \int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$
3. **Selecting $L_t\left(\boldsymbol{x}_t\right)$ to be a quadratic function with a constant $\int L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right) d\boldsymbol{x}_{t-1}$**; as a result, $\overline{\text{bel}}\left(\boldsymbol{x}_t\right) = \eta\, e^{-L_t(\boldsymbol{x}_t)}$
4. **Determining $L_t\left(\boldsymbol{x}_t\right)$ from L_t and $L_t\left(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\right)$**; this results in a quadratic function, which means that $\overline{\text{bel}}\left(\boldsymbol{x}_t\right)$ is a Gaussian distribution
   - The mean of the updated distribution is determined by the **minimum of $L_t\left(\boldsymbol{x}_t\right)$ over $\boldsymbol{x}_t$**
   - The covariance is determined by the **curvature of $L_t\left(\boldsymbol{x}_t\right)$, namely the inverse of the second derivative of $L_t\left(\boldsymbol{x}_t\right)$ with respect to $\boldsymbol{x}_t$**

Kalman Filter Derivation Sketch: Measurement Update

The derivation of the measurement update also involves multiple steps:

Kalman Filter Derivation Sketch: Measurement Update

The derivation of the measurement update also involves multiple steps:

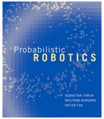
1. **Substituting** $p\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)$ **and** $\overline{\mathrm{bel}}\left(\boldsymbol{x}_{t-1}\right)$ **into** $\mathrm{bel}\left(\boldsymbol{x}_t\right)$; as both of them are Gaussian distributions; the belief is of the form $\mathrm{bel}\left(\boldsymbol{x}_t\right) = \eta e^{-J_t}$, where J_t is a quadratic function

Kalman Filter Derivation Sketch: Measurement Update

The derivation of the measurement update also involves multiple steps:

1. **Substituting $p\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)$ and $\overline{\text{bel}}\left(\boldsymbol{x}_{t-1}\right)$ into $\text{bel}\left(\boldsymbol{x}_t\right)$**; as both of them are Gaussian distributions; the belief is of the form $\text{bel}\left(\boldsymbol{x}_t\right) = \eta e^{-J_t}$, where J_t is a quadratic function
2. **Determining $\boldsymbol{\mu}_t$ by the minimum of the derivative of J_t**

Kalman Filter Derivation Sketch: Measurement Update



The derivation of the measurement update also involves multiple steps:

1. **Substituting** $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ **and** $\overline{\text{bel}}(\boldsymbol{x}_{t-1})$ **into** $\text{bel}(\boldsymbol{x}_t)$; as both of them are Gaussian distributions; the belief is of the form $\text{bel}(\boldsymbol{x}_t) = \eta e^{-J_t}$, where J_t is a quadratic function
2. **Determining** $\boldsymbol{\mu}_t$ **by the minimum of the derivative of** J_t
3. **Manipulating** K_t, which at this point is expressed as $K_t = \Sigma_t C_t^T Q_t^{-1}$, **so that the dependence on** Σ_t **is eliminated**

Kalman Filter Derivation Sketch: Measurement Update

The derivation of the measurement update also involves multiple steps:

1. **Substituting** $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ **and** $\overline{\text{bel}}(\boldsymbol{x}_{t-1})$ **into** $\text{bel}(\boldsymbol{x}_t)$; as both of them are Gaussian distributions; the belief is of the form $\text{bel}(\boldsymbol{x}_t) = \eta e^{-J_t}$, where J_t is a quadratic function
2. **Determining** $\boldsymbol{\mu}_t$ **by the minimum of the derivative of** J_t
3. **Manipulating** K_t, which at this point is expressed as $K_t = \Sigma_t C_t^T Q_t^{-1}$, **so that the dependence on** Σ_t **is eliminated**
4. **Calculating the covariance matrix** Σ_t **as the inverse of the second derivative of** J_t

Inversion Lemma (Sherman–Morrison–Woodbury Identity)

- The Kalman filter derivation makes use of a matrix identity called the **inversion lemma**
- The inversion lemma is expressed as

$$(A + PQR)^{-1} = A^{-1} - A^{-1}P\left(Q^{-1} + RA^{-1}P\right)^{-1}RA^{-1}$$

- Both the state update and the measurement update use this identity for rewriting expressions during the derivation

Kalman Filter and Non-Linear Models

- The linearity assumption of the Kalman filter is rather limiting — the motion and measurement models for a robot are often expressed by non-linear relations

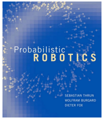
Kalman Filter and Non-Linear Models

- The linearity assumption of the Kalman filter is rather limiting — the motion and measurement models for a robot are often expressed by non-linear relations
- In this case, **the motion model is expressed by a non-linear function**

$$\boldsymbol{x}_t = g\left(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$ is process noise as before

Kalman Filter and Non-Linear Models



- The linearity assumption of the Kalman filter is rather limiting — the motion and measurement models for a robot are often expressed by non-linear relations

- In this case, **the motion model is expressed by a non-linear function**

$$\boldsymbol{x}_t = g\left(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) + \boldsymbol{\epsilon}_t$$

  where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$ is process noise as before

- Similarly, **the measurement model is now expressed by a non-linear function**

$$\boldsymbol{z}_t = h\left(\boldsymbol{x}_t\right) + \boldsymbol{\delta}_t$$

  where $\boldsymbol{\delta}_t \sim \mathcal{N}(0, Q_t)$ is again measurement noise

Kalman Filter and Non-Linear Models

- The linearity assumption of the Kalman filter is rather limiting — the motion and measurement models for a robot are often expressed by non-linear relations
- In this case, **the motion model is expressed by a non-linear function**

$$\boldsymbol{x}_t = g\left(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}\right) + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$ is process noise as before

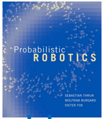
- Similarly, **the measurement model is now expressed by a non-linear function**

$$\boldsymbol{z}_t = h\left(\boldsymbol{x}_t\right) + \boldsymbol{\delta}_t$$

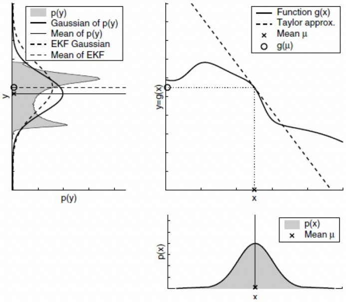
where $\boldsymbol{\delta}_t \sim \mathcal{N}(0, Q_t)$ is again measurement noise

- The Kalman filter needs to be extended so that it can be used in the non-linear case; we will now take a brief look at some extensions to make this possible

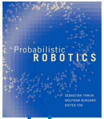
Extended Kalman Filter (EKF)

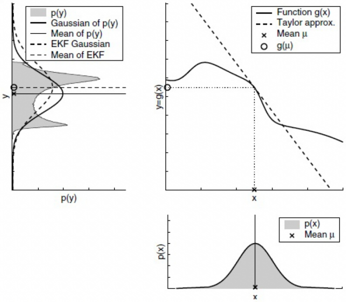


- The idea behind the extended Kalman filter is simple: **a first-order Taylor approximation of the non-linear models is performed** to eliminate the non-linearity



Extended Kalman Filter (EKF)





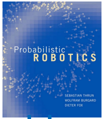
- The idea behind the extended Kalman filter is simple: **a first-order Taylor approximation of the non-linear models is performed** to eliminate the non-linearity

- **The motion model is linearised around μ_{t-1}:**

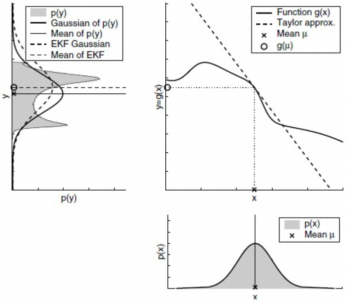
$$\begin{aligned} g(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) &\approx g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + g'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &= g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \end{aligned}$$

where G is the Jacobian of g

Extended Kalman Filter (EKF)



- The idea behind the extended Kalman filter is simple: **a first-order Taylor approximation of the non-linear models is performed** to eliminate the non-linearity



- **The motion model is linearised around μ_{t-1}:**

$$
\begin{aligned}
g(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) &\approx g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + g'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\
&= g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1})
\end{aligned}
$$

  where G is the Jacobian of g

- **The measurement model is linearised around $\overline{\mu}_t$:**

$$
\begin{aligned}
h(\boldsymbol{x}_t) &\approx h(\overline{\boldsymbol{\mu}}_t) + h'(\overline{\boldsymbol{\mu}}_t)(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t) \\
&= h(\overline{\boldsymbol{\mu}}_t) + H(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t)
\end{aligned}
$$

  where H is the Jacobian of h

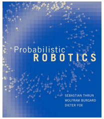
- These linearised estimates then **represent the means in the Gaussian motion and measurement models**

Extended Kalman Filter Summary

1. The state is updated based on the motion according to the non-linear motion model:

$$\overline{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$$

Extended Kalman Filter Summary



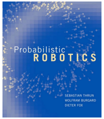
1. The state is updated based on the motion according to the non-linear motion model:

$$\boldsymbol{\overline{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$$

2. The covariance is updated based on the motion by considering the linearised motion model and the motion noise:

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R^t$$

Extended Kalman Filter Summary



1. The state is updated based on the motion according to the non-linear motion model:

$$\overline{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$$

2. The covariance is updated based on the motion by considering the linearised motion model and the motion noise:

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R^t$$

3. The Kalman gain is computed by considering the linearised measurement model, the updated covariance, and the measurement noise:

$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t \right)^{-1}$$

Extended Kalman Filter Summary

1. The state is updated based on the motion according to the non-linear motion model:

$$\overline{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$$

2. The covariance is updated based on the motion by considering the linearised motion model and the motion noise:

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R^t$$

3. The Kalman gain is computed by considering the linearised measurement model, the updated covariance, and the measurement noise:

$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t \right)^{-1}$$

4. The state and the covariance are updated based on the measurement

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t \left(\boldsymbol{z}_t - h\left(\overline{\boldsymbol{\mu}}_t \right) \right)$$

$$\Sigma_t = \left(I - K_t H_t \right) \overline{\Sigma}_t$$

Side-by-Side Comparison of the Linear and Extended Kalman Filters

Linear Kalman filter

$$\overline{\boldsymbol{\mu}}_t = A_t \boldsymbol{\mu}_{t-1} + B_t \boldsymbol{u}_t$$

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

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

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t \left(\boldsymbol{z}_t - C_t \overline{\boldsymbol{\mu}}_t \right)$$

$$\Sigma_t = \left(I - K_t C_t \right) \overline{\Sigma}_t$$

Extended Kalman filter

$$\overline{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$$

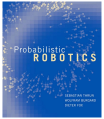
$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R^t$$

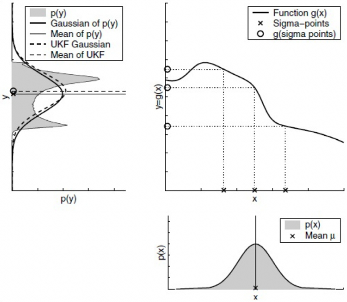
$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t \right)^{-1}$$

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t \left(\boldsymbol{z}_t - h \left(\overline{\boldsymbol{\mu}}_t \right) \right)$$

$$\Sigma_t = \left(I - K_t H_t \right) \overline{\Sigma}_t$$

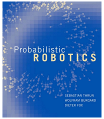
Unscented Kalman Filter (UKF)

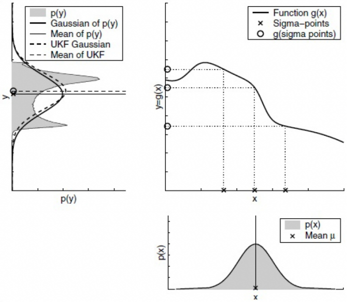




- The unscented Kalman filter performs linearisation using an **unscented transform**

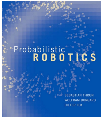
Unscented Kalman Filter (UKF)

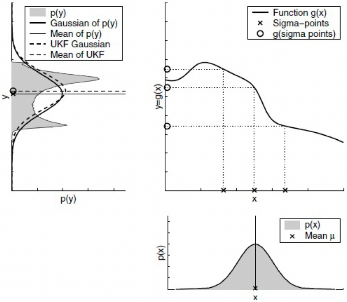




- The unscented Kalman filter performs linearisation using an **unscented transform**
- The idea behind the unscented transform is to approximate the non-linear distribution using **sigma points** selected from the distribution; these are **passed through the non-linear functions** to update the distribution's mean and covariance

Unscented Kalman Filter (UKF)





- The unscented Kalman filter performs linearisation using an **unscented transform**
- The idea behind the unscented transform is to approximate the non-linear distribution using **sigma points** selected from the distribution; these are **passed through the non-linear functions** to update the distribution's mean and covariance
- The unscented Kalman filter selects **sigma points from** $\text{bel}\left(x_{t-1}\right)$ **and** $\overline{\text{bel}}\left(x_t\right)$ to linearise the motion and measurement models

Unscented Transform

The transform selects $2n + 1$ **sigma points for an n-dimensional $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$** using the rule

$$\boldsymbol{p}_0 = \boldsymbol{\mu}$$
$$\boldsymbol{p}_i = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\Sigma}\right), 1 \leq i \leq n$$
$$\boldsymbol{p}_i = \boldsymbol{\mu} - \left(\sqrt{(n+\lambda)\Sigma}\right), n+1 \leq i \leq 2n$$

with $\lambda = \alpha^2(n + \kappa) - n$ — α and κ determine the locations of the points

Unscented Transform

The transform selects $2n + 1$ **sigma points for an n-dimensional $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$** using the rule

$$\boldsymbol{p}_0 = \boldsymbol{\mu}$$

$$\boldsymbol{p}_i = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\Sigma}\right), 1 \leq i \leq n$$

$$\boldsymbol{p}_i = \boldsymbol{\mu} - \left(\sqrt{(n+\lambda)\Sigma}\right), n+1 \leq i \leq 2n$$

with $\lambda = \alpha^2(n+\kappa) - n$ — α and κ determine the locations of the points

The mean and covariance are computed as **weighted averages over the sigma points**:

$$\boldsymbol{\mu}' = \sum_{i=0}^{2n} w_i^{\mu} g(\boldsymbol{p_i})$$

$$\Sigma' = \sum_{i=1}^{2n} w_i^{\Sigma} \left(g(\boldsymbol{p}_i) - \boldsymbol{\mu}'\right)\left(g(\boldsymbol{p}_i) - \boldsymbol{\mu}'\right)^T$$

Unscented Transform

The transform selects $2n+1$ **sigma points for an n-dimensional $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$** using the rule

$$\boldsymbol{p}_0 = \boldsymbol{\mu}$$
$$\boldsymbol{p}_i = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\Sigma}\right), 1 \leq i \leq n$$
$$\boldsymbol{p}_i = \boldsymbol{\mu} - \left(\sqrt{(n+\lambda)\Sigma}\right), n+1 \leq i \leq 2n$$

with $\lambda = \alpha^2(n+\kappa) - n$ — α and κ determine the locations of the points

The mean and covariance are computed as **weighted averages over the sigma points**:

$$\boldsymbol{\mu}' = \sum_{i=0}^{2n} w_i^{\mu} g(\boldsymbol{p_i})$$
$$\Sigma' = \sum_{i=1}^{2n} w_i^{\Sigma} \left(g(\boldsymbol{p}_i) - \boldsymbol{\mu}'\right)\left(g(\boldsymbol{p}_i) - \boldsymbol{\mu}'\right)^T$$

The weights are calculated as

$$w_0^{\mu} = \frac{\lambda}{n+\lambda}$$
$$w_0^{\Sigma} = \frac{\lambda}{n+\lambda}\left(1 - \alpha^2 + \beta\right)$$
$$w_i^{\mu} = w_i^{\Sigma} = \frac{1}{2(n+\lambda)}, 1 \leq i \leq 2n$$

Summary

- Localisation is the problem of determining a robot's pose in an environment
- The recursive Bayes filter is a family of algorithms that can be used for state estimation in general and localisation in particular
- The Kalman filter is one particular type of Bayes filter that represents the state by a Gaussian distribution and assumes linear motion and measurement models
- Extension of the Kalman filter, such as the extended Kalman Filter (EKF) and the unscented Kalman filter (UKF), can be used to deal with non-linear motion and measurement models