Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

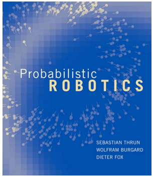
b-it Bonn-Aachen International Center for Information Technology

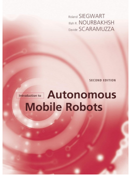
Institute for AI and Autonomous Systems

Simultaneous Localisation and Mapping (SLAM)

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure



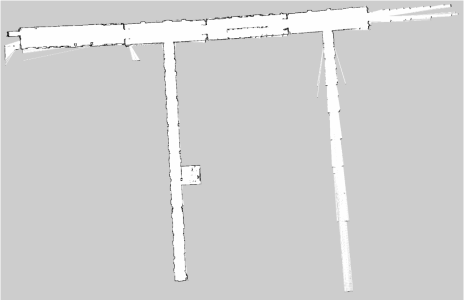


- Mapping preliminaries
- Occupancy grid mapping
- Simultaneous localisation and mapping (SLAM)
- SLAM algorithms

Mapping Preliminaries

What is Mapping?

- As discussed in our previous lectures, **a robot needs an environment representation so that it can act autonomously in an environment**
- A representation that enables autonomous navigation is referred to as a map — **the process of creating a map is called mapping**
- **Particularly for planar navigation, occupancy grids are quite commonly used**



A partial occupancy grid map of the second floor of the H-BRS C building

Challenges of Mapping

- The primary problem with mapping is that **creating a map requires the robot's location to be given at all times**, but **determining the location needs a map to be provided**
  - Mapping and localisation is thus a chicken-and-egg problem

Challenges of Mapping

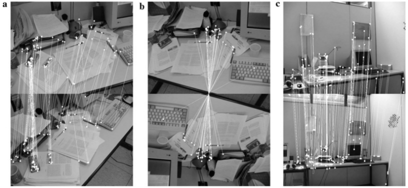
- The primary problem with mapping is that **creating a map requires the robot's location to be given at all times**, but **determining the location needs a map to be provided**
  - Mapping and localisation is thus a chicken-and-egg problem
- All mapping procedures involve processing perceptual features and representing those in a map, but **the quality and uniqueness of mapping depends on the types of features that are used for mapping**
  - Think about how people map environments — we need distinguishing features (e.g. flashy billboards) rather than unremarkable features (e.g. trees that all look the same)

Challenges of Mapping

- The primary problem with mapping is that **creating a map requires the robot's location to be given at all times**, but **determining the location needs a map to be provided**
  - Mapping and localisation is thus a chicken-and-egg problem
- All mapping procedures involve processing perceptual features and representing those in a map, but **the quality and uniqueness of mapping depends on the types of features that are used for mapping**
  - Think about how people map environments — we need distinguishing features (e.g. flashy billboards) rather than unremarkable features (e.g. trees that all look the same)
- A robot may also have multiple sensors and thus create multiple maps; in this case, there is a need to **combine maps that are based on multiple sensor modalities**

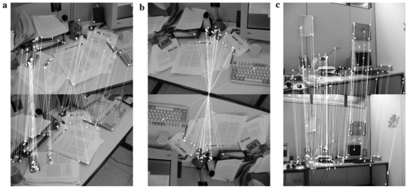
Mapping and Feature Correspondences

- Except in cases where perceptual features have unique IDs (e.g. unique QR codes), a **correspondence problem** needs to be resolved during mapping



Examples of visual feature correspondences. Taken from E. Delponte et al., "SVD-matching using SIFT features," *Graphical models*, vol. 68, no. 5-6, pp. 415–431, 2006.

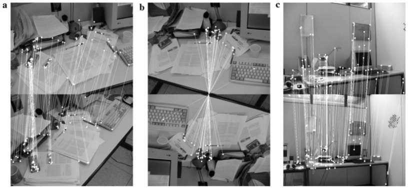
Mapping and Feature Correspondences



Examples of visual feature correspondences. Taken from E. Delponte et al., "SVD-matching using SIFT features," *Graphical models*, vol. 68, no. 5-6, pp. 415–431, 2006.

- Except in cases where perceptual features have unique IDs (e.g. unique QR codes), a **correspondence problem** needs to be resolved during mapping
- The correspondence problem is concerned with the identification of feature identities: **an observed feature should either be matched with known features or be identified as a new feature**

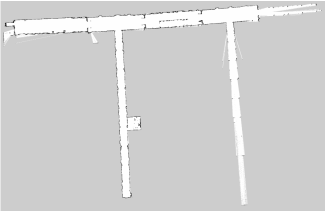
Mapping and Feature Correspondences



Examples of visual feature correspondences. Taken from E. Delponte et al., "SVD-matching using SIFT features," *Graphical models*, vol. 68, no. 5-6, pp. 415–431, 2006.

- Except in cases where perceptual features have unique IDs (e.g. unique QR codes), a **correspondence problem** needs to be resolved during mapping
- The correspondence problem is concerned with the identification of feature identities: **an observed feature should either be matched with known features or be identified as a new feature**
- This is a challenging **optimisation problem** — matching is typically performed by **minimising a distance metric to existing features**
  - The main challenge stems from **perceptual ambiguities** — distinct features may look indistinguishable from different views
  - **Computational cost** is also a challenge — the optimisation problem needs to be resolved continuously, as a robot moves around and collects new measurements

Mapping and Environment Representations



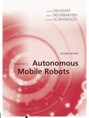
Occupancy grid created using a 2D laser

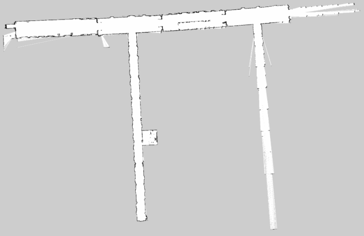
3D point cloud map created from RGB-D measurements. Taken from E. Sandström, Erik et al., "Point-SLAM: Dense Neural Point Cloud-based SLAM," in *Proc. IEEE/CVF Int. Conf. Computer Vision*, 2023.

- The exact manner in which mapping is performed strongly depends on the representation that is used to represent the map
  - Mapping is a very active research field, with new approaches being regularly proposed
- There are, however, various commonalities between different mapping techniques, which we attempt to discuss today
- We will start with the simplest type of mapping: 2D occupancy grid mapping

Occupancy Grid Mapping

Reminder: 2D Occupancy Grid





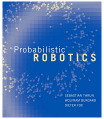
- A 2D occupancy grid is a two-dimensional discrete map that **represents an environment by a set of grid cells, each of which has an occupancy probability**
- We can define an occupancy grid to have the form $\mathcal{M}^{OC} = (w, h, r, o_x, o_y, OC)$, where:
  - r is the **grid cell resolution** (e.g. in meters)
  - w and h are the **map's width and height**, respectively
  - o_x and o_y represent the **coordinates of the map's origin point** (in continuous coordinates)
  - $OC \in \mathbb{R}^{h \times w}$ **is an occupancy probability matrix**, namely $OC_{i,j} \in [0, 1], 1 \leq i \leq h, 1 \leq j \leq w$
- 2D occupancy grid can be acquired from 2D measurements and are very commonly used in (indoor) robotics

Occupancy Grid Mapping Formalisation

- The overall objective of the occupancy grid mapping process is to **approximate the posterior occupancy distribution given the robot's movements and measurements**:

$$p(OC|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

Occupancy Grid Mapping Formalisation



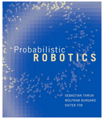
- The overall objective of the occupancy grid mapping process is to **approximate the posterior occupancy distribution given the robot's movements and measurements**:

$$p(OC|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

- A common assumption made in occupancy grid mapping is that **the occupancy probabilities are independent of each other**, in which case the map probability is written as

$$p(OC|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}) = \prod_{i=1}^{h} \prod_{j=1}^{w} p(OC_{i,j}|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

Occupancy Grid Mapping Formalisation



- The overall objective of the occupancy grid mapping process is to **approximate the posterior occupancy distribution given the robot's movements and measurements**:

$$p(OC|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

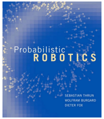
- A common assumption made in occupancy grid mapping is that **the occupancy probabilities are independent of each other**, in which case the map probability is written as

$$p(OC|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}) = \prod_{i=1}^{h} \prod_{j=1}^{w} p(OC_{i,j}|\boldsymbol{x}_{0:t}, \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

- Due to potential numerical instabilities, occupancy grids are often represented through the **log odds** instead of through the occupancy probabilities directly:

$$LOC_{i,j} = \ln \frac{OC_{i,j}}{1 - OC_{i,j}}$$

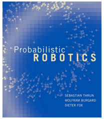
Occupancy Grid Mapping Algorithm



function OccupancyGridMapping(LOC, $\boldsymbol{x}_t$, $\boldsymbol{z}_t$)
  for $i \leftarrow 1$ **to** h **do**
    for $j \leftarrow 1$ **to** w **do**
      if $LOC_{i,j}$ in the field of $\boldsymbol{z}_t$ **then**
        $LOCi, j \leftarrow LOCi, j + \texttt{InverseSensorModel}((x_{LOC_{i,j}}, y_{LOC_{i,j}}), \boldsymbol{x}_t, \boldsymbol{z}_t) - l_0$
  return LOC

- Given the current robot's state $\boldsymbol{x}_t$ and a measurement $\boldsymbol{z}_t$, the idea behind occupancy grid mapping is to **update the occupancy probabilities of only those cells that fall within the sensor's field of view**

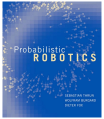
Occupancy Grid Mapping Algorithm



function $\textsc{OccupancyGridMapping}(LOC, \boldsymbol{x}_t, \boldsymbol{z}_t)$
  for $i \leftarrow 1$ **to** h **do**
    for $j \leftarrow 1$ **to** w **do**
      if $LOC_{i,j}$ in the field of $\boldsymbol{z}_t$ **then**
        $LOCi, j \leftarrow LOCi, j + \texttt{InverseSensorModel}((x_{LOC_{i,j}}, y_{LOC_{i,j}}), \boldsymbol{x}_t, \boldsymbol{z}_t) - l_0$
  return LOC

- Given the current robot's state $\boldsymbol{x}_t$ and a measurement $\boldsymbol{z}_t$, the idea behind occupancy grid mapping is to **update the occupancy probabilities of only those cells that fall within the sensor's field of view**
- Measurements are noisy! Because of this, the update of those cells uses an **inverse sensor model** $p(LOCi, j|\boldsymbol{z}_t)$

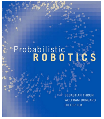
Occupancy Grid Mapping Algorithm



function $\textsc{OccupancyGridMapping}(LOC, \boldsymbol{x}_t, \boldsymbol{z}_t)$
  for $i \leftarrow 1$ **to** h **do**
    for $j \leftarrow 1$ **to** w **do**
      if $LOC_{i,j}$ in the field of $\boldsymbol{z}_t$ **then**
        $LOCi, j \leftarrow LOCi, j + \texttt{InverseSensorModel}((x_{LOC_{i,j}}, y_{LOC_{i,j}}), \boldsymbol{x}_t, \boldsymbol{z}_t) - l_0$
  return LOC

- Given the current robot's state $\boldsymbol{x}_t$ and a measurement $\boldsymbol{z}_t$, the idea behind occupancy grid mapping is to **update the occupancy probabilities of only those cells that fall within the sensor's field of view**
- Measurements are noisy! Because of this, the update of those cells uses an **inverse sensor model** $p(LOCi, j|\boldsymbol{z}_t)$
- **The algorithm updates the occupancies continuously, based on every new evidence** — dynamic obstacles that disappear quickly will thus not affect the mapping process significantly

Inverse Sensor Model

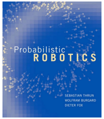


function $\textsc{InverseSensorModel}((x_{i,j}, y_{i,j}), (x_t, y_t, \theta_t), \boldsymbol{z}_t)$
    $r \leftarrow \sqrt{(x_{i,j} - x_t)^2 + (y_{i,j} - y_t)^2}$
    $\phi \leftarrow \text{atan2}(y_{i,j} - y_t, x_{i,j} - x_t) - \theta_t$
    $k \leftarrow \arg\min_j |\phi - \theta_{\boldsymbol{z}_j}|$
    if $r > \min(z_{\max}, \boldsymbol{z}_{t_k} + \alpha/2)$ **or** $|\phi - \theta_{\boldsymbol{z}_k}| > \beta/2$ **then**
        return l_0
    if $\boldsymbol{z}_t^k < z_{\max}$ **and** $|r - \boldsymbol{z}_t^k| < \alpha/2$ **then**
        return l_{occ}
    if $r \leq \boldsymbol{z}_{t_k}$ **then**
        return l_{free}

- The algorithm above is an **inverse sensor model for a range sensor** (e.g. a lidar), which returns the occupancy log odds:

$$l_{i,j} = \ln \frac{p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}{1 - p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}$$

Inverse Sensor Model



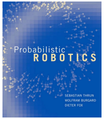
function $\textsc{InverseSensorModel}((x_{i,j}, y_{i,j}), (x_t, y_t, \theta_t), \boldsymbol{z}_t)$
    $r \leftarrow \sqrt{(x_{i,j} - x_t)^2 + (y_{i,j} - y_t)^2}$
    $\phi \leftarrow \text{atan2}(y_{i,j} - y_t, x_{i,j} - x_t) - \theta_t$
    $k \leftarrow \arg\min_j |\phi - \theta_{\boldsymbol{z}_j}|$
    if $r > \min(z_{\max}, \boldsymbol{z}_{t_k} + \alpha/2)$ **or** $|\phi - \theta_{\boldsymbol{z}_k}| > \beta/2$ **then**
        return l_0
    if $\boldsymbol{z}_t^k < z_{\max}$ **and** $|r - \boldsymbol{z}_t^k| < \alpha/2$ **then**
        return l_{occ}
    if $r \leq \boldsymbol{z}_{t_k}$ **then**
        return l_{free}

- The algorithm above is an **inverse sensor model for a range sensor** (e.g. a lidar), which returns the occupancy log odds:

$$l_{i,j} = \ln \frac{p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}{1 - p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}$$

- The above model is specified for an **angular range** β and a **distance tolerance** α

Inverse Sensor Model



function $\textsc{InverseSensorModel}((x_{i,j}, y_{i,j}), (x_t, y_t, \theta_t), \boldsymbol{z}_t)$

  $r \leftarrow \sqrt{(x_{i,j} - x_t)^2 + (y_{i,j} - y_t)^2}$

  $\phi \leftarrow \text{atan2}(y_{i,j} - y_t, x_{i,j} - x_t) - \theta_t$

  $k \leftarrow \arg\min_j |\phi - \theta_{\boldsymbol{z}_j}|$

  if $r > \min(z_{\max}, \boldsymbol{z}_{t_k} + \alpha/2)$ **or** $|\phi - \theta_{\boldsymbol{z}_k}| > \beta/2$ **then**

    return l_0

  if $\boldsymbol{z}_t^k < z_{\max}$ **and** $|r - \boldsymbol{z}_t^k| < \alpha/2$ **then**

    return l_{occ}

  if $r \leq \boldsymbol{z}_{t_k}$ **then**

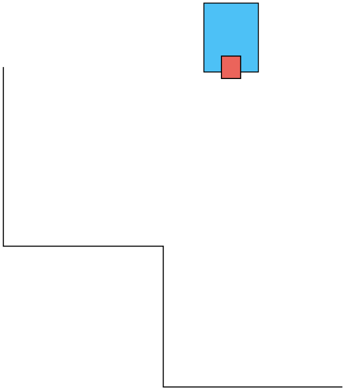
    return l_{free}

- The algorithm above is an **inverse sensor model for a range sensor** (e.g. a lidar), which returns the occupancy log odds:

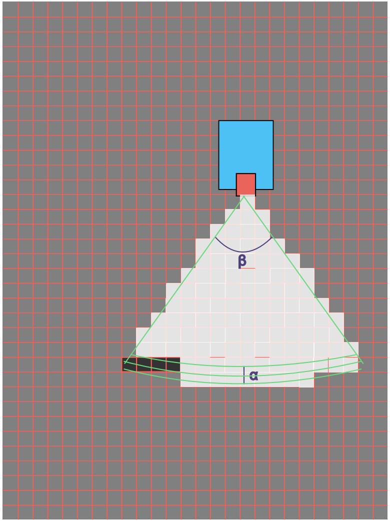
$$l_{i,j} = \ln \frac{p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}{1 - p(OC_{i,j} | \boldsymbol{x}_t, \boldsymbol{z}_t)}$$

- The above model is specified for an **angular range** β and a **distance tolerance** α
- In general, such an inverse sensor model can be learned

Occupancy Grid Mapping Illustration

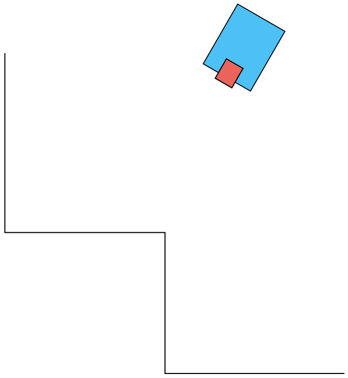


Real environment

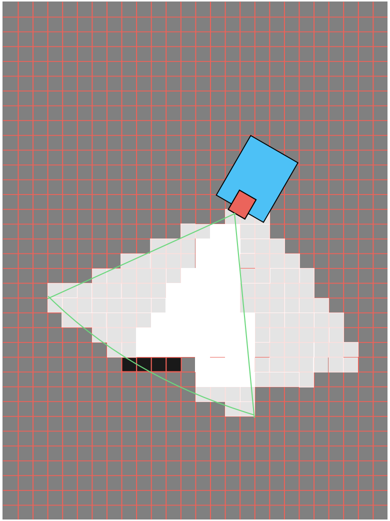


Mapping: $t = 1$

Occupancy Grid Mapping Illustration

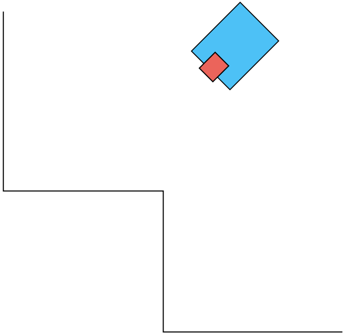


Real environment

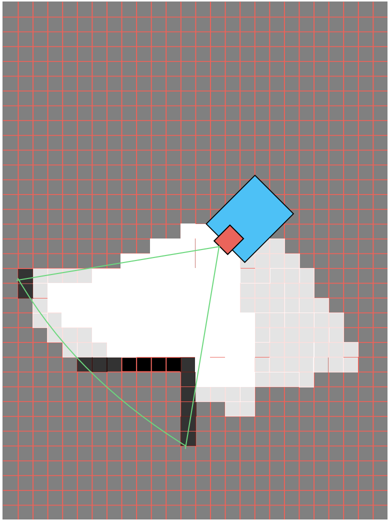


Mapping: $t = 2$

Occupancy Grid Mapping Illustration

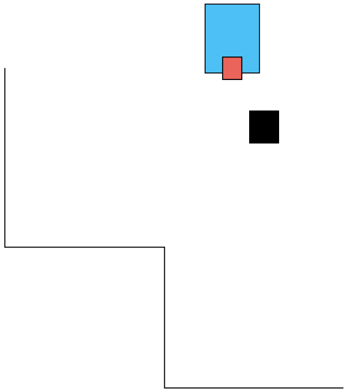


Real environment

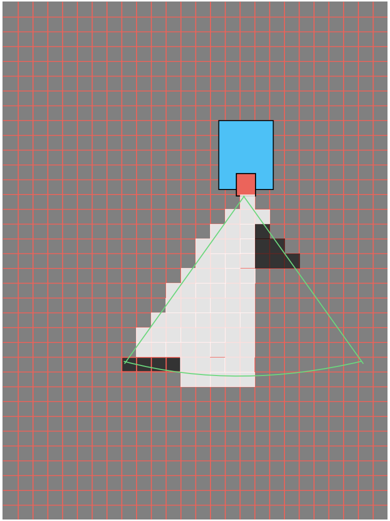


Mapping: $t = 3$

Occupancy Grid Mapping Illustration: Dynamic Obstacle

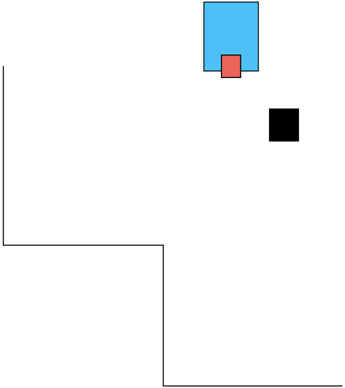


Real environment

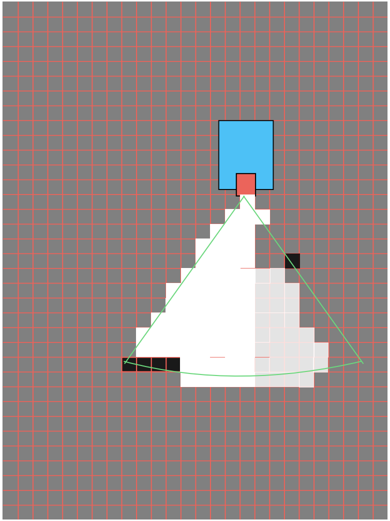


Mapping: $t = 1$

Occupancy Grid Mapping Illustration: Dynamic Obstacle

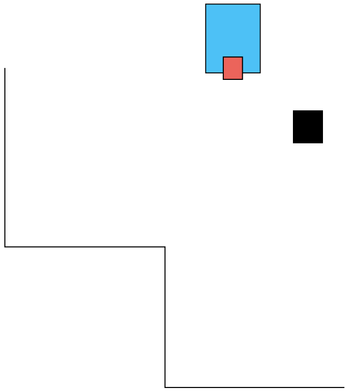


Real environment

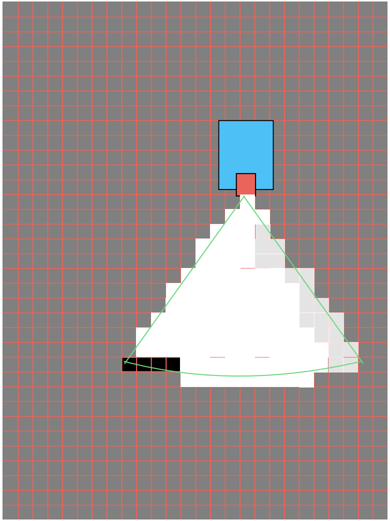


Mapping: $t = 2$

Occupancy Grid Mapping Illustration: Dynamic Obstacle

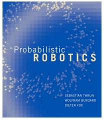


Real environment

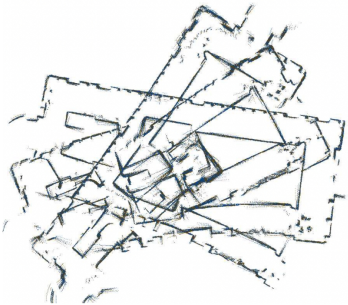


Mapping: $t = 3$

Occupancy Grid Mapping Challenges



- The main problem of this mapping technique is that **the poses are supposed to be known** — perfect, noise-free motion is thus assumed
- In reality, robot motions are noisy and will accumulate over time if not corrected — **this can lead a map to drift away**
  - The occupancy grid mapping algorithm does not perform **loop closure** — recognising that a landmark has previously been seen and correcting the estimate accordingly
- A more general technique is thus needed — one that enables a robot to **recover from the accumulation of motion errors**



Without loop closure and correction for a robot's noisy motion, an inaccurate map will be created

Simultaneous Localisation and Mapping (SLAM)

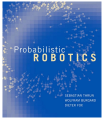
What is SLAM?

- SLAM is a technique based on which a robot can **create an environment map while determining the pose at the same time**
  - SLAM thus aims to resolve the main problem of mapping techniques that assume perfect robot motion

What is SLAM?

- SLAM is a technique based on which a robot can **create an environment map while determining the pose at the same time**
  - SLAM thus aims to resolve the main problem of mapping techniques that assume perfect robot motion
- The ultimate objective of SLAM is to **create an accurate environment map** that can subsequently be used for localisation
  - Thus, SLAM is a mapping technique at its core — the output of the procedure is the map, while the robot's pose estimates are typically discarded in the end

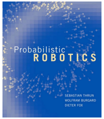
SLAM Formalisation



- The obvious way of defining SLAM is that of **computing the distribution of the current pose along with the map**, given all motion commands and measurements — referred to as the **online SLAM problem**:

$$p(\boldsymbol{x}_t, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

SLAM Formalisation



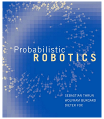
- The obvious way of defining SLAM is that of **computing the distribution of the current pose along with the map**, given all motion commands and measurements — referred to as the **online SLAM problem**:

$$p(\boldsymbol{x}_t, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

- In general, we may also be interested in the **distribution of the full path along with the map** — this is referred to as the **full SLAM problem**:

$$p(\boldsymbol{x}_{0:t}, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

SLAM Formalisation



- The obvious way of defining SLAM is that of **computing the distribution of the current pose along with the map**, given all motion commands and measurements — referred to as the **online SLAM problem**:

$$p(\boldsymbol{x}_t, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

- In general, we may also be interested in the **distribution of the full path along with the map** — this is referred to as the **full SLAM problem**:

$$p(\boldsymbol{x}_{0:t}, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$$

- SLAM is typically performed by **finding correspondences between landmarks**; considering correspondences explicitly, the SLAM problem can be defined as follows:
  - **Online SLAM Problem with correspondences**: $p(\boldsymbol{x}_t, c_t, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$
  - **Full SLAM Problem with correspondences**: $p(\boldsymbol{x}_{1:t}, c_{1:t}, M | \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t})$

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
    1. **A motion update is performed** based on a robot's motion command

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
  1. **A motion update is performed** based on a robot's motion command
  2. **Sensor measurements are collected**, and **feature extraction is performed**

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
  1. **A motion update is performed** based on a robot's motion command
  2. **Sensor measurements are collected**, and **feature extraction is performed**
  3. **Feature predictions are made** based on the robot's hypothesised location

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
  1. **A motion update is performed** based on a robot's motion command
  2. **Sensor measurements are collected**, and **feature extraction is performed**
  3. **Feature predictions are made** based on the robot's hypothesised location
  4. **A solution to the correspondence problem is found**

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
  1. **A motion update is performed** based on a robot's motion command
  2. **Sensor measurements are collected**, and **feature extraction is performed**
  3. **Feature predictions are made** based on the robot's hypothesised location
  4. **A solution to the correspondence problem is found**
  5. **The existing features are updated as necessary, or a new feature is recorded in the map**

How is SLAM Performed?

- There is no single way of performing SLAM, but, just like in the case of localisation, **many SLAM techniques are based on Bayesian filtering**
- A typical SLAM algorithm performs the following steps continuously:
  1. **A motion update is performed** based on a robot's motion command
  2. **Sensor measurements are collected**, and **feature extraction is performed**
  3. **Feature predictions are made** based on the robot's hypothesised location
  4. **A solution to the correspondence problem is found**
  5. **The existing features are updated as necessary, or a new feature is recorded in the map**
- SLAM algorithms differ in the way in which they perform the above steps

SLAM Challenges

- Unlike localisation, SLAM is a **high-dimensional problem** — the map features and feature correspondences need to be considered in addition to the pose

SLAM Challenges

- Unlike localisation, SLAM is a **high-dimensional problem** — the map features and feature correspondences need to be considered in addition to the pose
- SLAM also requires a solution to the **correspondence problem**, which is challenging in general — not just in the context of SLAM

SLAM Challenges

- Unlike localisation, SLAM is a **high-dimensional problem** — the map features and feature correspondences need to be considered in addition to the pose
- SLAM also requires a solution to the **correspondence problem**, which is challenging in general — not just in the context of SLAM
- The incorporation of new information into existing maps is also generally difficult — **dynamic mapping** may lead to deterioration of known maps

SLAM Algorithms

SLAM Techniques

- On the following slides, we will look at two major SLAM techniques:

EKF SLAM

A SLAM algorithm that uses an extended Kalman filter

FastSLAM

An algorithm using a Rao-Blackwellised particle filter (combining a particle filter with EKFs)

- We will also take a very brief look at **visual SLAM** — the currently predominant SLAM paradigm using visual information

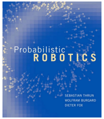
EKF SLAM

- In EKF SLAM, the state representation **combines the robot's pose with the positions and identities of the features in the map**; considering N **features** $m_i, 1 \leq i \leq N$ with **positions** (m_{i_x}, m_{i_y}) and **identities** m_{i_s}, the state in EKF SLAM is represented as

$$\boldsymbol{y}_t = \begin{pmatrix} \boldsymbol{x}_t & \boldsymbol{m} \end{pmatrix}^T = \begin{pmatrix} x & y & \theta & m_{1_x} & m_{1_y} & m_{1_s} & \dots & m_{N_x} & m_{N_y} & m_{N_s} \end{pmatrix}^T$$

- EKF SLAM can be performed relatively easily if the feature identities are known (e.g. if features have unique identities that can be easily identified)
- In the more general case of unknown feature identities, the algorithm needs to solve the correspondence problem

EKF SLAM with Known Correspondences



Algorithm EKF_SLAM_known_correspondences$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & \underbrace{0\cdots0}_{3N} \end{pmatrix}$$

$$\bar\mu_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

$$\bar\Sigma_t = G_t\,\Sigma_{t-1}\,G_t^T + F_x^T\,R_t\,F_x$$

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ *do*

$j = c_t^i$

if landmark j never seen before

$$\begin{pmatrix} \bar\mu_{j,x} \\ \bar\mu_{j,y} \\ \bar\mu_{j,s} \end{pmatrix} = \begin{pmatrix} \bar\mu_{t,x} \\ \bar\mu_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i\cos(\phi_t^i + \bar\mu_{t,\theta}) \\ r_t^i\sin(\phi_t^i + \bar\mu_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar\mu_{j,x} - \bar\mu_{t,x} \\ \bar\mu_{j,y} - \bar\mu_{t,y} \end{pmatrix}$$

$q = \delta^T\delta$

$$\hat z_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar\mu_{t,\theta} \\ \bar\mu_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$K_t^i = \bar\Sigma_t\,H_t^{iT}(H_t^i\,\bar\Sigma_t\,H_t^{iT} + Q_t)^{-1}$

$\bar\mu_t = \bar\mu_t + K_t^i(z_t^i - \hat z_t^i)$

$\bar\Sigma_t = (I - K_t^i\,H_t^i)\,\bar\Sigma_t$

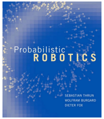
endfor

$\mu_t = \bar\mu_t$

$\Sigma_t = \bar\Sigma_t$

return μ_t, Σ_t

EKF SLAM with Known Correspondences



Algorithm EKF_SLAM_known_correspondences$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & \underbrace{0\cdots0}_{3N} \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

Motion update of the robot's pose

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ *do*

$j = c_t^i$

if landmark j *never seen before*

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i\ \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$q = \delta^T\delta$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$K_t^i = \bar{\Sigma}_t\,H_t^{iT}\,(H_t^i\,\bar{\Sigma}_t\,H_t^{iT} + Q_t)^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i(z_t^i - \hat{z}_t^i)$

$\bar{\Sigma}_t = (I - K_t^i\,H_t^i)\,\bar{\Sigma}_t$

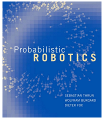
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Known Correspondences



Algorithm EKF_SLAM_known_correspondences$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N} \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

Motion update of the robot's pose

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i \ \phi_t^i \ s_t^i)^T$ *do*

$j = c_t^i$

if landmark j *never seen before*

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

Initialisation of a new feature's position

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{3j-3} & 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t \, H_t^{iT} (H_t^i \, \bar{\Sigma}_t \, H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i \, H_t^i) \, \bar{\Sigma}_t$$

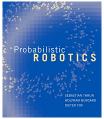
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Known Correspondences



Algorithm EKF_SLAM_known_correspondences($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t$):

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N} \end{pmatrix}$$

Motion update of the robot's pose

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i \; \phi_t^i \; s_t^i)^T$ *do*

$j = c_t^i$

Initialisation of a new feature's position

if landmark j never seen before

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

Measurement prediction

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{3j-3} & 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t \, H_t^{iT} (H_t^i \, \bar{\Sigma}_t \, H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i \, H_t^i) \, \bar{\Sigma}_t$$

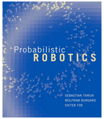
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Known Correspondences



Algorithm EKF_SLAM_known_correspondences($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t$):

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & \underbrace{0\cdots0}_{3N} \end{pmatrix}$$

Motion update of the robot's pose

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ do

$j = c_t^i$

Initialisation of a new feature's position

if landmark j never seen before

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i\cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i\sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

Measurement prediction

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T\delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

Measurement update

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t\,H_t^{iT}(H_t^i\,\bar{\Sigma}_t\,H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i(z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i\,H_t^i)\,\bar{\Sigma}_t$$

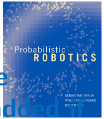
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Unknown Correspondences



In the unknown correspondence case, **each observed feature is either matched to one of the known features** — for instance, using maximum likelihood estimation — or **a new feature is added if all known features exceed a distance threshold**

Algorithm EKF_SLAM$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, N_{t-1})$:

$N_t = N_{t-1}$

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \ \phi_t^i\ \ s_t^i)^T$ *do*

$$\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

for $k = 1$ *to* N_t+1 *do*

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$q_k = \delta_k^T\,\delta_k$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \operatorname{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$$

$$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$H_t^k = \frac{1}{q_k} \begin{pmatrix} -\sqrt{q}_k\delta_{k,x} & -\sqrt{q}_k\delta_{k,y} & 0 & \sqrt{q}_k\delta_{k,x} & \sqrt{q}_k\delta_{k,y} & 0 \\ \delta_{k,y} & -\delta_{k,x} & -1 & -\delta_{k,y} & \delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$$

$\Psi_k = H_t^k\,\bar{\Sigma}_t\,[H_t^k]^T + Q_t$

$\pi_k = (z_t^i - \hat{z}_t^k)^T\,\Psi_k^{-1}\,(z_t^i - \hat{z}_t^k)$

endfor

$\pi_{N_t+1} = \alpha$

$j(i) = \underset{k}{\operatorname{argmin}}\ \pi_k$

$N_t = \max\{N_t, j(i)\}$

$K_t^i = \bar{\Sigma}_t\,[H_t^{j(i)}]^T\Psi_{j(i)}^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i\,(z_t^i - \hat{z}_t^{j(i)})$

$\bar{\Sigma}_t = (I - K_t^i\,H_t^{j(i)})\,\bar{\Sigma}_t$

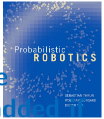
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Unknown Correspondences



In the unknown correspondence case, **each observed feature is either matched to one of the known features** — for instance, using maximum likelihood estimation — or **a new feature is added if all known features exceed a distance threshold**

Algorithm EKF_SLAM($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, N_{t-1}$):

$N_t = N_{t-1}$

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ *do*

$$\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

for $k = 1$ *to* N_t+1 *do*

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$q_k = \delta_k^T \delta_k$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \operatorname{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$$

$$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$H_t^k = \frac{1}{q_k} \begin{pmatrix} -\sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & \sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} & 0 \\ \delta_{k,y} & -\delta_{k,x} & -1 & -\delta_{k,y} & \delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$$

$\Psi_k = H_t^k\, \bar{\Sigma}_t\, [H_t^k]^T + Q_t$

$\pi_k = (z_t^i - \hat{z}_t^k)^T\, \Psi_k^{-1}\, (z_t^i - \hat{z}_t^k)$

Similarity calculation

endfor

$\pi_{N_t+1} = \alpha$

$j(i) = \underset{k}{\operatorname{argmin}}\; \pi_k$

$N_t = \max\{N_t, j(i)\}$

$K_t^i = \bar{\Sigma}_t\, [H_t^{j(i)}]^T \Psi_{j(i)}^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i\, (z_t^i - \hat{z}_t^{j(i)})$

$\bar{\Sigma}_t = (I - K_t^i\, H_t^{j(i)})\, \bar{\Sigma}_t$

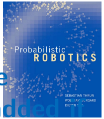
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

EKF SLAM with Unknown Correspondences



In the unknown correspondence case, **each observed feature is either matched to one of the known features** — for instance, using maximum likelihood estimation — or **a new feature is added if all known features exceed a distance threshold**

Algorithm EKF_SLAM($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, N_{t-1}$):

$N_t = N_{t-1}$

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

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

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ *do*

$$\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

for $k = 1$ *to* N_t+1 *do*

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$q_k = \delta_k^T \delta_k$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \text{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$$

$$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$H_t^k = \frac{1}{q_k} \begin{pmatrix} -\sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & \sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} & 0 \\ \delta_{k,y} & -\delta_{k,x} & -1 & -\delta_{k,y} & \delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$$

$\Psi_k = H_t^k\,\bar{\Sigma}_t\,[H_t^k]^T + Q_t$

$\pi_k = (z_t^i - \hat{z}_t^k)^T\,\Psi_k^{-1}\,(z_t^i - \hat{z}_t^k)$

Similarity calculation

endfor

$\pi_{N_t+1} = \alpha$

$j(i) = \underset{k}{\text{argmin}}\;\pi_k$

$N_t = \max\{N_t, j(i)\}$

Similarity maximisation to existing features or adding a new feature

$K_t^i = \bar{\Sigma}_t\,[H_t^{j(i)}]^T \Psi_{j(i)}^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i\,(z_t^i - \hat{z}_t^{j(i)})$

$\bar{\Sigma}_t = (I - K_t^i\,H_t^{j(i)})\,\bar{\Sigma}_t$

endfor

$\mu_t = \bar{\mu}_t$

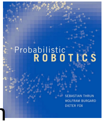
$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

Problems with EKF SLAM

- Just as in the localisation case, only a single hypothesis is pursued by EKF SLAM — **additional recovery procedures need to be implemented to recover from a localisation loss**
- With a large number of features, EKF SLAM needs to work with large matrices (in the order of the number of features N) — **the algorithm can thus be memory and computationally expensive**
- The algorithm **does not consider features that have not been observed, but which should have been under the pose hypothesis**

FastSLAM



- **FastSLAM is a particle filter-based SLAM algorithm** that solves the full SLAM problem
- As in EKF SLAM, **the state representation combines the robot's path and the states of features**, such that **each feature is represented by an extended Kalman filter**
  - Thus, each feature $\boldsymbol{m}_i, 1 \leq i \leq N$ has an associated Gaussian distribution $\boldsymbol{\mu}_i, \Sigma_i$
- Each particle $\boldsymbol{p}^j, 1 \leq j \leq M$ in FastSLAM is represented as follows:

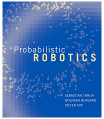
$$\boldsymbol{p}^j = \left[(x_1, y_1, \theta_1), \ldots, (x_t, y_t, \theta_t), (\boldsymbol{\mu}_1, \Sigma_1), \ldots, (\boldsymbol{\mu}_N, \Sigma_N), w^j\right]$$

- **FastSLAM is based on an insight that poses and features are conditionally independent** given the control updates, measurements, and correspondences, so the posterior distribution is given as

$$p(\boldsymbol{y}_{1:t}|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{c}_{1:t}) = p(\boldsymbol{x}_{1:t}|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{c}_{1:t}) \prod_{i=1}^{N} p(\boldsymbol{m}_i|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{c}_{1:t})$$

  This factorisation is known as Rao-Blackwellisation, and thus the particle filter as a **Rao-Blackwellised particle filter**

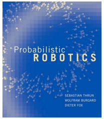
FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do	// loop over all particles
retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}	
$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$	// sample pose
$j = c_t$	// observed feature
if feature j never seen before	
$\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$	// initialize mean
$H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$	// calculate Jacobian
$\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$	// initialize covariance
$w^{[k]} = p_0$	// default importance weight
else	
$\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$	// measurement prediction
$H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$	// calculate Jacobian
$Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$	// measurement covariance
$K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$	// calculate Kalman gain
$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$	// update mean
$\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$	// update covariance
$w^{[k]} = \lvert 2\pi Q \rvert^{-\frac{1}{2}}\, \exp\left\{ -\frac{1}{2}(z_t - \hat{z}_n)^T\, Q^{-1}\, (z_t - \hat{z}_n) \right\}$	// importance factor
endif	
for all other features $j' \neq j$ do	// unobserved features
$\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$	// leave unchanged
$\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$	
endfor	
endfor	
$Y_t = \emptyset$	// initialize new particle set
do M times	// resample M particles
draw random k with probability $\propto w^{[k]}$	// resample
add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t	
endfor	
return Y_t	

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // loop over all particles

retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}

$\boxed{x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t) \quad \text{// sample pose}}$

Motion update of the particle's pose

$j = c_t$ // observed feature

if feature j never seen before

$\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // initialize mean

$H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // calculate Jacobian

$\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // initialize covariance

$w^{[k]} = p_0$ // default importance weight

else

$\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // measurement prediction

$H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // calculate Jacobian

$Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // measurement covariance

$K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // calculate Kalman gain

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // update mean

$\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // update covariance

$w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1}\, (z_t - \hat{z}_n)\right\}$ // importance factor

endif

for all other features $j' \neq j$ do // unobserved features

$\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // leave unchanged

$\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$

endfor

endfor

$Y_t = \emptyset$ // initialize new particle set

do M times // resample M particles

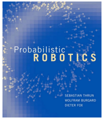
draw random k with probability $\propto w^{[k]}$ // resample

add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t

endfor

return Y_t

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // loop over all particles
  retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}
  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // sample pose

Motion update of the particle's pose

  $j = c_t$ // observed feature
  if feature j never seen before
    $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // initialize mean
    $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // calculate Jacobian
    $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // initialize covariance
    $w^{[k]} = p_0$ // default importance weight

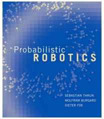
Initialisation of a new feature's position

  else
    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // measurement prediction
    $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // calculate Jacobian
    $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // measurement covariance
    $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // calculate Kalman gain
    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // update mean
    $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // update covariance
    $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n)\right\}$ // importance factor
  endif
  for all other features $j' \neq j$ do // unobserved features
    $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // leave unchanged
    $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$
  endfor
endfor

$Y_t = \emptyset$ // initialize new particle set
do M times // resample M particles
  draw random k with probability $\propto w^{[k]}$ // resample
  add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t
endfor

return Y_t

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // loop over all particles
  retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}

  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // sample pose

Motion update of the particle's pose

  $j = c_t$ // observed feature
  if feature j never seen before
    $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // initialize mean
    $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // calculate Jacobian
    $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // initialize covariance
    $w^{[k]} = p_0$ // default importance weight

Initialisation of a new feature's position

  else
    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // measurement prediction
    $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // calculate Jacobian
    $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // measurement covariance
    $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // calculate Kalman gain
    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // update mean
    $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // update covariance

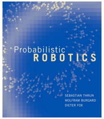
Position update of an existing feature

    $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n)\right\}$ // importance factor
  endif
  for all other features $j' \neq j$ do // unobserved features
    $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // leave unchanged
    $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$
  endfor
endfor

$Y_t = \emptyset$ // initialize new particle set
do M times // resample M particles
  draw random k with probability $\propto w^{[k]}$ // resample
  add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t
endfor

return Y_t

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // loop over all particles
  retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}
  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // sample pose
  $j = c_t$ // observed feature
  if feature j never seen before
    $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // initialize mean
    $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // calculate Jacobian
    $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // initialize covariance
    $w^{[k]} = p_0$ // default importance weight
  else
    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // measurement prediction
    $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // calculate Jacobian
    $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // measurement covariance
    $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // calculate Kalman gain
    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // update mean
    $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // update covariance
    $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n)\right\}$ // importance factor
  endif
  for all other features $j' \neq j$ do // unobserved features
    $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // leave unchanged
    $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$
  endfor
endfor

$Y_t = \emptyset$ // initialize new particle set
do M times // resample M particles
  draw random k with probability $\propto w^{[k]}$ // resample
  add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t
endfor

return Y_t

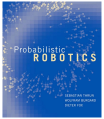
Motion update of the particle's pose

Initialisation of a new feature's position

Position update of an existing feature

Weight update based on the measurement / prediction discrepancy

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // *loop over all particles*
  retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}
  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample pose*
  $j = c_t$ // *observed feature*
  if feature j never seen before
    $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*
    $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // *calculate Jacobian*
    $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // *initialize covariance*
    $w^{[k]} = p_0$ // *default importance weight*
  else
    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*
    $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // *calculate Jacobian*
    $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // *measurement covariance*
    $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // *calculate Kalman gain*
    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // *update mean*
    $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // *update covariance*
    $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n)\right\}$ // *importance factor*
  endif
  for all other features $j' \neq j$ do // *unobserved features*
    $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // *leave unchanged*
    $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$
  endfor
endfor

$Y_t = \emptyset$ // *initialize new particle set*
do M times // *resample M particles*
  draw random k with probability $\propto w^{[k]}$ // *resample*
  add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t
endfor

return Y_t

Motion update of the particle's pose

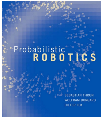
Initialisation of a new feature's position

Position update of an existing feature

Weight update based on the measurement / prediction discrepancy

Unobserved features are not updated

FastSLAM with Known Correspondences



Algorithm FastSLAM 1.0_known_correspondence(z_t, c_t, u_t, Y_{t-1}):

for $k = 1$ to M do // *loop over all particles*

  retrieve $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}

  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample pose*

  $j = c_t$ // *observed feature*

  if feature j never seen before

    $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*

    $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ // *calculate Jacobian*

    $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // *initialize covariance*

    $w^{[k]} = p_0$ // *default importance weight*

  else

    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*

    $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ // *calculate Jacobian*

    $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // *measurement covariance*

    $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // *calculate Kalman gain*

    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ // *update mean*

    $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // *update covariance*

    $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q^{-1} (z_t - \hat{z}_n)\right\}$ // *importance factor*

  endif

  for all other features $j' \neq j$ do // *unobserved features*

    $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ // *leave unchanged*

    $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$

  endfor

endfor

$Y_t = \emptyset$ // *initialize new particle set*

do M times // *resample M particles*

  draw random k with probability $\propto w^{[k]}$ // *resample*

  add $\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_N^{[k]}, \Sigma_N^{[k]} \right\rangle \right\rangle$ to Y_t

endfor

return Y_t

Motion update of the particle's pose

Initialisation of a new feature's position

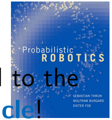
Position update of an existing feature

Weight update based on the measurement / prediction discrepancy

Unobserved features are not updated

Particle resampling

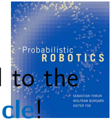
FastSLAM with Unknown Correspondences



As in EKF SLAM, if the correspondences are not given, each observed feature is either matched to the known features or a new feature is added; this operation needs to be performed **for every particle**!

Algorithm FastSLAM 1.0(z_t, u_t, Y_{t-1}):

for $k = 1$ to M do // *loop over all particles*
  retrieve $\left\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]},t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}
  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample new pose*
  for $j = 1$ to $N_{t-1}^{[k]}$ do // *measurement likelihoods*
    $\hat{z}_j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*
    $H_j = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*
    $Q_j = H_j\, \Sigma_{j,t-1}^{[k]}\, H_j^T + Q_t$ // *measurement covariance*
    $w_j = |2\pi Q_j|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_j)^T Q_j^{-1}(z_t - \hat{z}_j)\right\}$ // *likelihood of correspondence*
  endfor
  $w_{1+N_{t-1}^{[k]}} = p_0$ // *importance factor, new feature*
  $w^{[k]} = \max w_j$ // *max likelihood correspondence*
  $\hat{c} = \operatorname{argmax} w_j$ // *index of ML feature*
  $N_t^{[k]} = \max\{N_{t-1}^{[k]}, \hat{c}\}$ // *new number of features in map*
  for $j = 1$ to $N_t^{[k]}$ do // *update Kalman filters*
    if $j = \hat{c} = 1 + N_{t-1}^{[k]}$ then // *is new feature?*
      $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*
      $H_j = h'(\mu_{j,t}^{[k]}, x_t^{[k]});\ \Sigma_{j,t}^{[k]} = (H_j^{-1})^T Q_t H_j^{-1}$ // *initialize covar.*
      $i_{j,t}^{[k]} = 1$ // *initialize counter*
    else if $j = \hat{c} \le N_{t-1}^{[k]}$ then // *is observed feature?*
      $K = \Sigma_{j,t-1}^{[k]} H_j^T Q_{\hat{c}}^{-1}$ // *calculate Kalman gain*
      $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}_{\hat{c}})$ // *update mean*
      $\Sigma_{j,t}^{[k]} = (I - K\, H_j)\Sigma_{j,t-1}^{[k]}$ // *update covariance*
      $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ // *increment counter*
    else // *all other features*
      $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ // *copy old mean*
      $\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]}$ // *copy old covariance*
      if $\mu_{j,t-1}^{[k]}$ outside perceptual range of $x_t^{[k]}$ then // *should feature have been seen?*
        $i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ // *no, do not change*
      else
        $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ // *yes, decrement counter*
        if $i_{j,t-1}^{[k]} < 0$ then
          discard feature j // *discard dubious features*
        endif
      endif
    endif
  endfor
  add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ to Y_{aux}
endfor
$Y_t = \emptyset$ // *construct new particle set*
do M times // *resample M particles*
  draw random index k with probability $\propto w^{[k]}$ // *resample*
  add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ to Y_t
enddo
return Y_t



FastSLAM with Unknown Correspondences

As in EKF SLAM, if the correspondences are not given, each observed feature is either matched to the known features or a new feature is added; this operation needs to be performed **for every particle**!

Algorithm FastSLAM 1.0(z_t, u_t, Y_{t-1}):

for $k = 1$ to M do // loop over all particles

retrieve $\left\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]},t-1}^{[k]} \right\rangle \right\rangle$ from Y_{t-1}

$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // sample new pose

for $j = 1$ to $N_{t-1}^{[k]}$ do // measurement likelihoods

$\hat{z}_j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // measurement prediction

$H_j = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // calculate Jacobian

$Q_j = H_j\, \Sigma_{j,t-1}^{[k]}\, H_j^T + Q_t$ // measurement covariance

$w_j = |2\pi Q_j|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(z_t - \hat{z}_j)^T Q_j^{-1}(z_t - \hat{z}_j) \right\}$ // likelihood of correspondence

endfor

Similarity calculation

$w_{1+N_{t-1}^{[k]}} = p_0$ // importance factor, new feature

$w^{[k]} = \max w_j$ // max likelihood correspondence

$\hat{c} = \operatorname{argmax} w_j$ // index of ML feature

$N_t^{[k]} = \max\{N_{t-1}^{[k]}, \hat{c}\}$ // new number of features in map

for $j = 1$ to $N_t^{[k]}$ do // update Kalman filters

if $j = \hat{c} = 1 + N_{t-1}^{[k]}$ then // is new feature?

$\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // initialize mean

$H_j = h'(\mu_{j,t}^{[k]}, x_t^{[k]});\ \Sigma_{j,t}^{[k]} = (H_j^{-1})^T Q_t H_j^{-1}$ // initialize covar.

$i_{j,t}^{[k]} = 1$ // initialize counter

else if $j = \hat{c} \le N_{t-1}^{[k]}$ then // is observed feature?

$K = \Sigma_{j,t-1}^{[k]} H_j^T Q_{\hat{c}}^{-1}$ // calculate Kalman gain

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}_{\hat{c}})$ // update mean

$\Sigma_{j,t}^{[k]} = (I - K\, H_j)\Sigma_{j,t-1}^{[k]}$ // update covariance

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ // increment counter

else // all other features

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ // copy old mean

$\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]}$ // copy old covariance

if $\mu_{j,t-1}^{[k]}$ outside perceptual range of $x_t^{[k]}$ then // should feature have been seen?

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ // no, do not change

else

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ // yes, decrement counter

if $i_{j,t-1}^{[k]} < 0$ then

discard feature j // discard dubious features

endif

endif

endif

endfor

add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ to Y_{aux}

endfor

$Y_t = \emptyset$ // construct new particle set

do M times // resample M particles

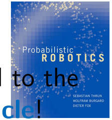
draw random index k with probability $\propto w^{[k]}$ // resample

add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ to Y_t

enddo

return Y_t

FastSLAM with Unknown Correspondences



As in EKF SLAM, if the correspondences are not given, each observed feature is either matched to the known features or a new feature is added; this operation needs to be performed **for every particle**!

Algorithm FastSLAM 1.0(z_t, u_t, Y_{t-1}):

for $k = 1$ *to* M *do* // *loop over all particles*

retrieve $\left\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]},t-1}^{[k]} \right\rangle \right\rangle$ *from* Y_{t-1}

$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample new pose*

for $j = 1$ *to* $N_{t-1}^{[k]}$ *do* // *measurement likelihoods*

$\hat{z}_j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*

$H_j = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

$Q_j = H_j\, \Sigma_{j,t-1}^{[k]}\, H_j^T + Q_t$ // *measurement covariance*

$w_j = |2\pi Q_j|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_j)^T Q_j^{-1}(z_t - \hat{z}_j)\right\}$ // *likelihood of correspondence*

endfor

$w_{1+N_{t-1}^{[k]}} = p_0$ // *importance factor, new feature*

$w^{[k]} = \max w_j$ // *max likelihood correspondence*

$\hat{c} = \operatorname{argmax} w_j$ // *index of ML feature*

$N_t^{[k]} = \max\{N_{t-1}^{[k]}, \hat{c}\}$ // *new number of features in map*

for $j = 1$ *to* $N_t^{[k]}$ *do* // *update Kalman filters*

if $j = \hat{c} = 1 + N_{t-1}^{[k]}$ *then* // *is new feature?*

$\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*

$H_j = h'(\mu_{j,t}^{[k]}, x_t^{[k]}); \Sigma_{j,t}^{[k]} = (H_j^{-1})^T Q_t H_j^{-1}$ // *initialize covar.*

$i_{j,t}^{[k]} = 1$ // *initialize counter*

else if $j = \hat{c} \leq N_{t-1}^{[k]}$ *then* // *is observed feature?*

$K = \Sigma_{j,t-1}^{[k]} H_j^T Q_{\hat{c}}^{-1}$ // *calculate Kalman gain*

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}_{\hat{c}})$ // *update mean*

$\Sigma_{j,t}^{[k]} = (I - K\, H_j)\Sigma_{j,t-1}^{[k]}$ // *update covariance*

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ // *increment counter*

else // *all other features*

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ // *copy old mean*

$\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]}$ // *copy old covariance*

if $\mu_{j,t-1}^{[k]}$ *outside perceptual range of* $x_t^{[k]}$ *then* // *should feature have been seen?*

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ // *no, do not change*

else

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ // *yes, decrement counter*

if $i_{j,t-1}^{[k]} < 0$ *then*

discard feature j // *discard dubious features*

endif

endif

endif

endfor

add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ *to* Y_{aux}

endfor

$Y_t = \emptyset$ // *construct new particle set*

do M *times* // *resample* M *particles*

draw random index k *with probability* $\propto w^{[k]}$ // *resample*

add $\left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle$ *to* Y_t

enddo

return Y_t

Similarity calculation

Similarity maximisation to existing features or registering a new feature

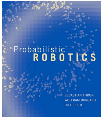
Problems with FastSLAM

- A basic implementation of the algorithm is inefficient and has time complexity of $O(MN)$ — this is because the procedure **performs inefficient updates on all features** — a more efficient implementation can make use of shared feature memory structures that reduce the time complexity

Problems with FastSLAM

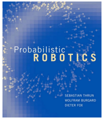
- A basic implementation of the algorithm is inefficient and has time complexity of $O(MN)$ — this is because the procedure **performs inefficient updates on all features** — a more efficient implementation can make use of shared feature memory structures that reduce the time complexity
- FastSLAM discards features that are considered spurious, but this can introduce **difficulties with loop closure** if particles with (past) path segments that are responsible for those measurements are discarded in the resampling process

Problems with FastSLAM



- A basic implementation of the algorithm is inefficient and has time complexity of $O(MN)$ — this is because the procedure **performs inefficient updates on all features** — a more efficient implementation can make use of shared feature memory structures that reduce the time complexity
- FastSLAM discards features that are considered spurious, but this can introduce **difficulties with loop closure** if particles with (past) path segments that are responsible for those measurements are discarded in the resampling process
- In general — and just as in the localisation case — **a diverse set of particles should be maintained** to prevent too early convergence on an incorrect map estimate

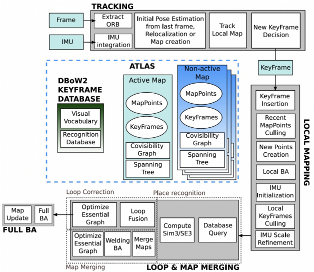
FastSLAM with Occupancy Grids



- The FastSLAM algorithm can be used with occupancy grid maps as well — unlike EKF SLAM, which would not be able to deal with the computational complexity of large grids
- In this case, instead of maintaining map features in the representation, **each particle maintains an occupancy grid map**
  - The map of each particle is then updated after performing a motion command and collecting an associated measurement
  - The weight of each particle is calculated based on the pose and measurement likelihood in the associated map
- The resulting algorithm is referred to as **grid-based FastSLAM**
  - The popular **gmapping** package in ROS is an implementation of this algorithm

Visual SLAM

- While 2D occupancy grids are commonly used in robot applications, **visual maps are conceptually desirable and interesting for robotics** — visual data is a considerably richer information source than range sensors



C. Campos et al., "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," in *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

- **Visual SLAM** is a family of techniques that use visual data (RGB or RGB-D images); many visual SLAM approaches exist in the literature[1,2]
  - Visual information is often combined with inertial measurements for improved motion tracking; this is referred to as **visual-inertial SLAM**
- A large variety of feature descriptors are applied in visual SLAM, e.g. SIFT, SURF, or ORB; loop closure can be performed using **place recognition**, which can be done using a **bag-of-visual-words**

- Note that **EKF SLAM and FastSLAM are conceptually agnostic to the map / feature representation**; thus, they can also be used in visual SLAM

[1] I. A. Kazerouni et al., "A survey of state-of-the-art on visual SLAM," *Expert Systems with Applications*, vol. 205, Nov. 2022.
[2] A. Macario Barros A et al., "A Comprehensive Survey of Visual SLAM Algorithms," *Robotics*, vol. 11, no. 1, Feb. 2022.

Summary

- Mapping is a process of creating an environment representation from sensor data, but the mapping process depends on accurate ground-truth localisation
- Occupancy grid mapping is a Bayesian filtering procedure that acquires an occupancy grid representation from range sensor measurements
- SLAM is a process of creating a map by also estimating a robot's pose; the problem is concerned with finding an estimate of the distribution of maps and a robot's pose (online SLAM) or a robot's path (full SLAM)
- There is a large variety of SLAM algorithms; mirroring our localisation discussion, we concretely looked at EKF SLAM, which is an algorithm based on extended Kalman filters, and FastSLAM, which uses a Rao-Blackwellised particle filter
- Visual SLAM uses data from cameras (exclusively or combined with other data sources) to create a visual 3D map
- Solving the feature correspondence problem is an essential component of all SLAM algorithms