AA274A: Principles of Robot Autonomy I
Course Notes

The previous chapter introduced the robot localization problem, but assumed that the map $\boldsymbol{m}$ was *given*. However, in many real-world robotics applications a map might not be known ahead of time, and therefore it wold need to be built on-the-fly. This problem, which involves using information about measurements $\boldsymbol{z}$ and controls $\boldsymbol{u}$ to simultaneously localize the robot in the world and build a map, is known as *simultaneous localization and mapping* (SLAM).

18 Simultaneous Localization and Mapping (SLAM)

Many real-world settings are challenging for robotic autonomy because both the map and the relative pose of the robot are unknown. For example, such a situation would occur in autonomous search-and-rescue operations where a robot needs to explore an unknown environment. The SLAM problem addresses this challenge by estimating the robot pose and constructing a map of the environment at the same time, based only on measurement $\boldsymbol{z}_{1:t}$ and control $\boldsymbol{u}_{1:t}$ data.

Generally speaking there are two types of SLAM problems that can be considered. The *online* SLAM problem aims to estimate the posterior $p(\boldsymbol{x}_t, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$ over the robot's current pose $\boldsymbol{x}_t$ and the map $\boldsymbol{m}$. Alternatively, the *full* SLAM problem estimates the entire path of the robot instead of just the current position, namely $p(\boldsymbol{x}_{1:t}, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$. The difference between these two SLAM problems is demonstrated graphically in Figure 1. Both SLAM problems experience the same challenge: error in the pose causes error in map estimation and error in map estimation causes error in the pose estimate. In this chapter, algorithms for both the online and full SLAM problems are studied.

18.1 EKF SLAM Algorithm

One of the earliest approaches to the online SLAM problem leverages the extended Kalman filter, and is essentially an extension of the EKF localization algorithm discussed in the previous chapter. Again, the key aspects to the approach are the exploitation of Gaussian distributions to model the robot's belief distribution $bel(\boldsymbol{x}_t)$, and state transition and measurement models. It will also be assumed that the map is feature-based:

$$\boldsymbol{m} = \{m_1, m_2, \dots, m_N\},$$

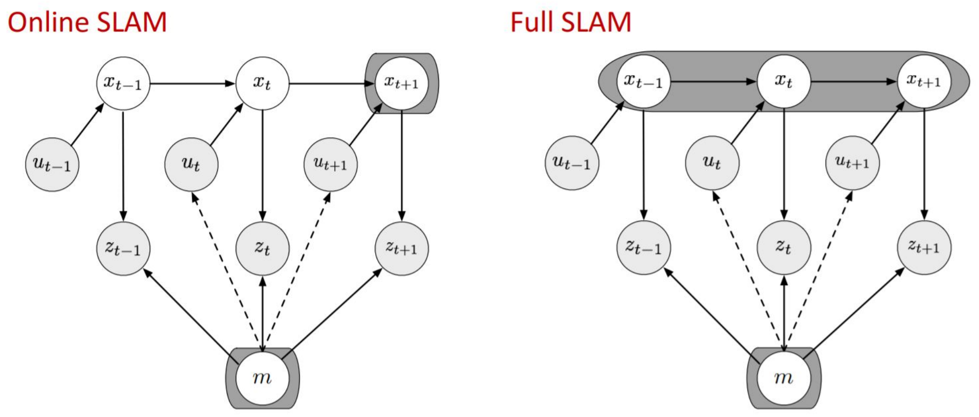

Figure 1: Difference between online and full SLAM, where online SLAM only estimates the current robot pose while full SLAM also estimates the robot's history.

where m_i is the i-th landmark with coordinates $(m_{i,x}, m_{i,y})$. As in the EKF localization problem, the measurement correspondences can either be assumed to be known or unknown (more common in practice).

The main idea behind EKF SLAM is that the coordinates $(m_{i,x}, m_{i,y})$ of each landmark m_i are added, along with the robot pose $\boldsymbol{x}_t$, to an augmented state vector:

$$\boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{m}_1 \\ \vdots \\ \boldsymbol{m}_N \end{bmatrix}, \tag{1}$$

where $\boldsymbol{m}_i = [m_{i,x}, m_{i,y}]^T$. With the new state vector $\boldsymbol{y}$ the online SLAM problem is to compute the posterior:

$$bel(\boldsymbol{y}_t) = p(\boldsymbol{y}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}).$$

EKF SLAM approaches have the advantage of being computationally efficient such that they can be run online, and are also well understood from a theoretical perspective. They can also provide good performance when the uncertainty is low. However, their main disadvantages are that they are restricted by the Gaussian assumption to unimodal estimates, and that performance can degrade in settings with high uncertainty or when the states are not well approximated by normal distributions.

18.1.1 State Transition and Measurement Models

Assuming that the landmarks $m_i \in \boldsymbol{m}$ are static, the state transition model for the augmented state vector $\boldsymbol{y}$ is assumed to be given by:

$$\boldsymbol{y}_t = g(\boldsymbol{u}_t, \boldsymbol{y}_{t-1}) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_t),$$

where the nonlinear vector function g is defined by:

$$g(\boldsymbol{u}_t, \boldsymbol{y}_{t-1}) = \begin{bmatrix} \tilde{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \\ \boldsymbol{m}_{1,t-1} \\ \vdots \\ \boldsymbol{m}_{N,t-1} \end{bmatrix},$$

and $\tilde{g}$ is the original robot motion model (e.g. differential drive robot model). The noise covariance is also defined as:

$$\boldsymbol{R}_t = \begin{bmatrix} \tilde{\boldsymbol{R}}_t & 0 \\ 0 & 0 \end{bmatrix},$$

where $\tilde{\boldsymbol{R}}_t$ is the noise covariance associated with the original robot motion model and the rest of the matrix are zeros. The Jacobian of the augmented motion model is defined as $G_t := \nabla_{\boldsymbol{y}} g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$ where $\boldsymbol{\mu}_{t-1}$ is the expected value of the belief distribution $bel(\boldsymbol{y}_{t-1})$ at the previous time.

The measurement model is defined in the same way as the previous chapter:

$$\boldsymbol{z}_t^i = h(\boldsymbol{y}_t, j) + \boldsymbol{\delta}_t,$$

where $\boldsymbol{\delta}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t)$ is Gaussian zero-mean noise and j is the index of the map feature $m_j \in \boldsymbol{m}$ that measurement i is associated with. The Jacobian is also defined in the same way with $H_t^j = \nabla_{\boldsymbol{y}} h(\bar{\boldsymbol{\mu}}_t, j)$, where $\bar{\boldsymbol{\mu}}_t$ is the predicted mean (that results from the EKF prediction step) of the distribution $\overline{bel}(\boldsymbol{y}_t)$.

18.1.2 EKF SLAM with Known Correspondences

As was the case in EKF localization, it is important to specify whether the the correspondences c_t^i between the i-th measurement $\boldsymbol{z}_t^i$ and the associated landmark in the map is known. In this section an EKF SLAM algorithm will be developed which assumes the correspondences $\boldsymbol{c}_t = [c_t^1, \dots]^T$ are *known*.

Algorithm 1 presents the EKF SLAM algorithm with known correspondences. It is almost identical to the EKF localization algorithm from last chapter, except that the state vector is augmented with the landmark positions and the positions of these landmarks are initialized when they are first seen. For this algorithm a general initialization of the belief distribution $bel(\boldsymbol{y}_0)$ is with:

$$\boldsymbol{\mu}_0 = \begin{bmatrix} \boldsymbol{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_0 = \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_0 & 0 & \cdots & 0 \\ 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \infty \end{bmatrix},$$

where:

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \tilde{\boldsymbol{\Sigma}}_0 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$$

```
Data: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{c}_t$
Result: $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$
$\bar{\boldsymbol{\Sigma}}_t = G_t \boldsymbol{\Sigma}_{t-1} G_t^T + \boldsymbol{R}_t$
foreach $\boldsymbol{z}_t^i$ do
    $j = c_t^i$
    if landmark $j$ never seen before then
        Initialize $\begin{bmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{bmatrix}$ as expected position based on $\boldsymbol{z}_t^i$
    end
    $S_t^i = H_t^j \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T + \boldsymbol{Q}_t$
    $K_t^i = \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T [S_t^i]^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_t^i(\boldsymbol{z}_t^i - h(\bar{\boldsymbol{\mu}}_t, j))$
    $\bar{\boldsymbol{\Sigma}}_t = (I - K_t^i H_t^j)\bar{\boldsymbol{\Sigma}}_t$
end
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t$
return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
```

Algorithm 1: Extended Kalman Filter Online SLAM Algorithm

and $\boldsymbol{x}_0$ and $\tilde{\boldsymbol{\Sigma}}$ are the initial robot state and associated covariance (which is set to zero). Since the reference frame for the map can be defined arbitrarily, this initialization is used to say that the initial robot pose is known to be at the origin with certainty (and the map is built with respect to that origin). The covariance of the map positions is set to infinity to reflect that there is initially no knowledge of their position.

18.2 EKF SLAM with Unknown Correspondences

Performing EKF SLAM when the correspondences between measurements and landmarks are *unknown* poses a more challenging problem. In the EKF localization case (when the map was known), a maximum likelihood method was used to determine correspondence. A similar approach is taken for EKF SLAM, which uses a maximum likelihood approach based on the *estimated* landmark positions. The main difference is that now a mechanism for hypothesizing that a new landmark has been found is also required. The EKF SLAM with unknown correspondences algorithm is given in Algorithm 2.

As can be seen there are a couple differences between Algorithm 1 and Algorithm 2. First, the measurements $\boldsymbol{z}_k^i$ are used to *hypothesize* the position of a new landmark. The Mahalanobis distance d_t^{ik} is then computed for all currently tracked landmarks, and the hypothesized landmark is added if the distance exceeds a threshold α (i.e. $d_t^{ik} > \alpha$ for all $k = 1, \dots, N_t$).

While this EKF-based algorithm can be used to solve the online SLAM problem without correspondences, it is not necessarily the most robust approach. In particular, extraneous

```
Data: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, N_{t-1}$
Result: $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
$N_t = N_{t-1}$
$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$
$\bar{\boldsymbol{\Sigma}}_t = G_t \boldsymbol{\Sigma}_{t-1} G_t^T + \boldsymbol{R}_t$
foreach $\boldsymbol{z}_t^i$ do
    Hypothesize position $\begin{bmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \end{bmatrix}$ from $\boldsymbol{z}_t^i$
    foreach $k = 1$ to $N_t + 1$ do
        $\hat{\boldsymbol{z}}_t^k = h(\bar{\boldsymbol{\mu}}_t, k)$
        $S_t^k = H_t^k \bar{\boldsymbol{\Sigma}}_t [H_t^k]^T + \boldsymbol{Q}_t$
        $d_t^{ik} = (\boldsymbol{z}_t^i - \hat{\boldsymbol{z}}_t^k)^T [S_t^k]^{-1} (\boldsymbol{z}_t^i - \hat{\boldsymbol{z}}_t^k)$
    end
    $d_t^{i(N_t+1)} = \alpha$
    $j = \arg\min_k \; d_t^{ik}$
    $N_t = \max\{N_t, j\}$
    $K_t^i = \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T [S_t^j]^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_t^i (\boldsymbol{z}_t^i - \hat{\boldsymbol{z}}_t^j)$
    $\bar{\boldsymbol{\Sigma}}_t = (I - K_t^i H_t^j) \bar{\boldsymbol{\Sigma}}_t$
end
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t$
return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
```

Algorithm 2: EKF Online SLAM Algorithm, Unknown Correspondences

measurements can result in the creation of fake landmarks, which will then propagate forward to future steps and cannot be corrected! There are several techniques to mitigate these issues, such as using outlier rejection schemes or strategies to enhance the distinctiveness of landmarks (which may require prior knowledge or assumptions). Another important disadvantage of EKF SLAM is that its computational complexity is quadratic with the number of landmarks N, but generally a large number of landmarks is required for good localization accuracy!

Example 18.1 (Differential Drive Robot with Range and Bearing Measurements)**.** Consider a differential drive robot with state $\boldsymbol{x} = [x, y, \theta]^T$, and suppose a sensor is available on the robot which measures the range r and bearing ϕ of landmarks $m_j \in \boldsymbol{m}$ relative to the robot's local coordinate frame. Additionally, multiple measurements corresponding to different features can be collected at each time step:

$$\boldsymbol{z}_t = \{[r_t^1, \phi_t^1]^T, [r_t^2, \phi_t^2]^T, \dots\},$$

where each measurement $\boldsymbol{z}_t^i$ contains the range r_t^i and bearing ϕ_t^i.

For the SLAM problem, the augmented state $\boldsymbol{y}_t$ is defined as:

$$\boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{m}_1 \\ \vdots \\ \boldsymbol{m}_N \end{bmatrix} = \begin{bmatrix} x & y & \theta & m_{1,x} & m_{1,y} & \dots & m_{N,x} & m_{N,y} \end{bmatrix}^T.$$

Assuming the correspondences are known, the measurement model for the range and bearing is:

$$h(\boldsymbol{y}_t, j) = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{bmatrix}. \tag{2}$$

The measurement Jacobian H_t^j corresponding to a measurement from landmark j is then given by:

$$H_t^j = \begin{bmatrix} -\frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_{t,j}}} & -\frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_{t,j}}} & 0 & 0 & \dots & 0 & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_{t,j}}} & \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_{t,j}}} & 0 & \dots \\ \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_{t,j}} & -\frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_{t,j}} & -1 & 0 & \dots & 0 & -\frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_{t,j}} & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_{t,j}} & 0 & \dots \end{bmatrix}, \tag{3}$$

where:

$$q_{t,j} := (\bar{\mu}_{j,x} - \bar{\mu}_{t,x})^2 + (\bar{\mu}_{j,y} - \bar{\mu}_{t,y})^2,$$

and $\bar{\mu}_{j,x}$ and $\bar{\mu}_{j,y}$ are the estimate of the x and y coordinates of landmark m_j from $\bar{\boldsymbol{\mu}}_t$.

With both a range and bearing measurement, the *expected* position of landmark m_j is given by:

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

This can be used in the known-correspondence EKF SLAM algorithm (Algorithm 1) to initialize the landmark position and can be used in the unknown-correspondence case (Algorithm 2) to hypothesize the position of new landmarks.

18.3 Particle SLAM Algorithm

Another approach to the robot SLAM problem is to leverage the non-parametric particle filter. In fact, particle SLAM can be used to solve the *full* SLAM problem, unlike EKF SLAM which only solves the online SLAM problem. Specifically, the full SLAM problem is to estimate the posterior distribution $p(\boldsymbol{x}_{1:t}, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$, which includes the full robot path $\boldsymbol{x}_{1:t}$ up to time t and the map $\boldsymbol{m}$. Similar to the EKF SLAM case, the robot state $\boldsymbol{x}_{1:t}$ and map feature positions $\boldsymbol{m}$ are combined into an augmented state vector $\boldsymbol{y}_{1:t}$ as in (1).

A naïve implementation of the particle filter in the context of full SLAM would be computationally intractable, since the number of particles required to belief distribution would be extremely large. However, the key insight that makes this approach tractable is that the posterior over the map elements is *conditionally independent* given the true path of the robot. Therefore the mapping component to the problem can be split up into separate problems, corresponding to each feature in the map! Splitting the problem in this way makes the overall problem much easier to solve.

Overall, particle filter SLAM approaches can be used with *any* noise distribution and can express multimodal beliefs since they are non-parametric. Additionally, in practice they can be relatively easy to implement and can also be more robust to data association errors. Their main disadvantages are that they typically do not scale well to large scale problems (too many particles are needed), and that without enough particles convergence may not occur.

18.3.1 Factoring the Posterior

The key insight of particle SLAM that makes it a computationally tractable algorithm is that the posterior $p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ can be factored as:

$$p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) = p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}), \tag{4}$$

where $\boldsymbol{m}_n$ is the n-th feature in the map $\boldsymbol{m}$, the term $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ is referred to as the *path posterior*, and the terms $p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t})$ are referred to as the *feature posteriors*.

This factorization can be derived by first using Bayes' rule

$$p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) = p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}),$$

and then noting that since the feature posterior is conditioned on $\boldsymbol{x}_{1:t}$, the dependence on $\boldsymbol{u}_{1:t}$ is redundant:

$$p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) = p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}).$$

Now the feature posterior $p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t})$ can be explored in more detail. In particular two cases can be considered for each landmark $\boldsymbol{m}_n$: the case when the measurement at time t is not associated with n and the case when it is:

$$p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}) = \begin{cases} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1}), & c_t \neq n, \\ \frac{p(\boldsymbol{z}_t \mid \boldsymbol{m}_n, \boldsymbol{x}_t, c_t) p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1})}{p(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t-1}, c_{1:t})}, & c_t = n, \end{cases}$$

where in the second case Bayes' rule was applied. It is now possible to show the result (4) by induction. First, suppose that:

$$p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1}) = \prod_{n=1}^{N} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1}).$$

Then, using Bayes' rule at time t:

$$\begin{aligned} p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}) &= \frac{p(\boldsymbol{z}_t \mid \boldsymbol{m}, \boldsymbol{x}_t, c_t) p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1})}{p(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t-1}, c_{1:t})}, \\ &= \frac{p(\boldsymbol{z}_t \mid \boldsymbol{m}, \boldsymbol{x}_t, c_t)}{p(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t-1}, c_{1:t})} \prod_{n=1}^{N} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}, c_{1:t-1}). \end{aligned}$$

Next, applying the analysis above for the cases where $c_t \neq n$ and $c_t = n$:

$$\begin{aligned} p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}) &= p(\boldsymbol{m}_{c_t} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}) \prod_{n \neq c_t} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}), \\ &= \prod_{n=1}^{N} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}). \end{aligned}$$

18.3.2 Fast SLAM with Known Correspondences

The particle SLAM algorithm referred to as Fast SLAM uses the factorization of the posterior $p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ in (4) to decompose the full SLAM problem into more manageable subproblems. Specifically, the path posterior $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ is estimated using a particle filter and the feature posteriors $p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t})$ are estimated by EKFs conditioned on the robot path $\boldsymbol{x}_{1:t}$ (i.e. there is a separate EKF for each feature $\boldsymbol{m}_n$).

Accordingly, the set of particles is given as:

$$\mathcal{Y}_t := \{Y_t^{[1]}, Y_t^{[2]}, ..., Y_t^{[M]}\},$$

where the k-th particle is defined by:

$$Y_t^{[k]} = \{\boldsymbol{x}_t^{[k]}, \boldsymbol{\mu}_{1,t}^{[k]}, \boldsymbol{\Sigma}_{1,t}^{[k]}, \dots, \boldsymbol{\mu}_{N,t}^{[k]}, \boldsymbol{\Sigma}_{N,t}^{[k]}\},$$

where $\boldsymbol{x}_t^{[k]}$ is a hypothesis of the robot state at time t, $(\boldsymbol{\mu}_{n,t}^{[k]}, \boldsymbol{\Sigma}_{n,t}^{[k]})$ are the mean and covariance of the EKF associated with landmark $\boldsymbol{m}_n$, and where it is assumed that there are N total landmarks in the map $\boldsymbol{m}$. As can be seen, with a total of M particles there are a total of NM EKFs! To summarize, the Fast SLAM algorithm is a particle based algorithm where each particle keeps track of a hypothesis of the robot state as well as the location (and uncertainty) of each landmark in the map! The algorithm is defined in Algorithm 3.

Note the blending of the classical particle filter algorithm with the EKF localization algorithm. In particular, the particle filter steps can be seen with the sampling of the new pose $\boldsymbol{x}_t$ from the state transition model and the use of the weights w for resampling a new set of particles (i.e. the measurement correction step). The EKF portions of the algorithm correspond to how the features are tracked, and in particular how the mean and covariance of the Gaussian corresponding to each landmark are updated based on new measurements.

References

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics.* MIT press, 2005.

```
**Data:** $\mathcal{Y}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, c_t$
**Result:** $\mathcal{Y}_t$
**for** $k = 1$ **to** $M$ **do**
    Sample $\boldsymbol{x}_t^{[k]} \sim p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}^{[k]})$
    $j = c_t$
    **if** *landmark $j$ never seen before* **then**
        Initialize feature: $(\boldsymbol{\mu}_{j,t-1}^{[k]}, \boldsymbol{\Sigma}_{j,t-1}^{[k]})$
    **else**
        $\hat{\boldsymbol{z}}^{[k]} = h(\boldsymbol{\mu}_{j,t-1}^{[k]}, \boldsymbol{x}_t^{[k]})$
        $S = H^j \boldsymbol{\Sigma}_{j,t-1}^{[k]} [H^j]^T + \boldsymbol{Q}_t$
        $K = \boldsymbol{\Sigma}_{j,t-1}^{[k]} [H^j]^T [S]^{-1}$
        $\boldsymbol{\mu}_{j,t}^{[k]} = \boldsymbol{\mu}_{j,t-1}^{[k]} + K(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})$
        $\boldsymbol{\Sigma}_{j,t}^{[k]} = (I - KH^j)\boldsymbol{\Sigma}_{j,t-1}^{[k]}$
        $w^{[k]} = \det(2\pi S)^{-1/2} \exp\big(-\frac{1}{2}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})\boldsymbol{Q}^{-1}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})\big)$
    **end**
    **for** $n \in \{1, \dots, N\}$, $n \neq c_t$ **do**
        $\boldsymbol{\mu}_{n,t}^{[k]} = \boldsymbol{\mu}_{n,t-1}^{[k]}$
        $\boldsymbol{\Sigma}_{n,t}^{[k]} = \boldsymbol{\Sigma}_{n,t-1}^{[k]}$
    **end**
**end**
$\mathcal{Y}_t = \emptyset$
**for** $m = 1$ **to** $M$ **do**
    Draw $k$ with probability $\propto w_t^{[k]}$
    $\mathcal{Y}_t = \mathcal{Y}_t \cup (\bar{\boldsymbol{x}}_t^{[k]}, \boldsymbol{\mu}_{1,t}^{[k]}, \boldsymbol{\Sigma}_{1,t}^{[k]}, \dots, \boldsymbol{\mu}_{N,t}^{[k]}, \boldsymbol{\Sigma}_{N,t}^{[k]})$
**end**
**return** $\mathcal{Y}_t$
```

Algorithm 3: Fast SLAM Algorithm