16 Robot Localization

The last few chapters introduced some of the most widely used algorithms based on Bayes' filter for probabilistic robot localization and state estimation. However these fundamental algorithms still need further enhancements before application to many robot localization tasks, since in their standard form they don't incorporate a notion of a local *map*. For example, a particle filter could be applied in its original form to a problem of global localization based on GNSS measurements, but localizing based on range measurements requires knowledge about *what* object is being ranged, and *where* that object is with respect to the local environment (i.e. the map). In this chapter a more specific definition of mobile robot localization is considered[1], namely the problem of determining the pose of a robot relative to a *given map* of the environment.

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005

Robot Localization

Localization with respect to a map can be interpreted as a problem of coordinate transformation. Maps are described in a global coordinate system, which is independent of a robot's pose. Localization can then be viewed as the process of establishing a correspondence between the map coordinate system and the robot's local coordinate system. Knowing this coordinate transformation then enables the robot to express the location of objects of interest within its own coordinate frame (a necessary prerequisite for robot autonomy).

In 2D problems, knowing the pose $\boldsymbol{x}_t = [x, y, \theta]^\top$ of a robot is sufficient to establish this correspondence, and an ideal sensor would directly be able to measure this pose. However in practice no such sensor exists, and therefore *indirect* (often noisy) measurements $\boldsymbol{z}_t$ of the pose are used. Since it is almost impossible to be able to reliably estimate $\boldsymbol{x}_t$ from a single measurement $\boldsymbol{z}_t$, localization algorithms typically integrate additional data *over time* to build reliable localization estimates. For example, consider a robot located inside a building where many corridors look alike. In this case a single sensor measurement (e.g. a range scan) is usually insufficient to disambiguate the identity of the corridor from the others.

In this chapter it will be seen how this map-based localization problem can be cast in the Bayesian filtering framework, such that the algorithms from previous chapters can be leveraged.

16.1 A Taxonomy of Localization Problems

To understand the broad scope of challenges related to robot localization, it is useful to develop a brief taxonomy of localization problems. This categorization will divide localization problems along a number of important dimensions pertaining to the nature of the environment (e.g. static versus dynamic), the initial knowledge that a robot may possess, and how information about the environment is gathered (e.g. passive or active, with one robot or collaboratively with several robots).

16.1.1 Local vs. Global

Localization problems can be characterized by the type of knowledge that is available initially, which has a significant impact on what type of localization algorithm is most appropriate for the problem.

- *Position tracking* problems assume that the initial pose of the robot is known. In these types of problems only incremental updates are required (i.e. the localization error is generally always small), and therefore unimodal Gaussian filters (e.g. Kalman filters) can be efficiently applied.

- *Global localization* problems assume that the initial pose of the robot is unknown. In these scenarios the use of a unimodal parametric belief distribution cannot adequately capture the global uncertainty. Therefore it is more appropriate to use non-parametric, multi-hypothesis filters, such as the particle filter.

- The *kidnapped robot problem* is a variant of the global localization problem (i.e. unknown initial pose) where the robot can get "kidnapped" and "teleported" to some other location. This problem is more difficult than the global localization problem since the localization algorithm needs to have an awareness that sudden drastic to the robot's pose are possible. While robots are typically not "kidnapped" in practice, the consideration of this type of problem is useful for ensuring the localization algorithm is *robust*, since the ability to recover from failure is essential for truly autonomous robots. Similar to the global localization problem, these problems are often best addressed using non-parametric, multi-hypothesis filters.

16.1.2 Static vs. Dynamic

Environmental changes are another important consideration in mobile robot localization, specifically whether they are static or dynamic.

- In *static* environments the robot is the only object that moves. Static environments are generally much easier to perform localization in.

- *Dynamic* environments possess objects other than the robot whose locations or configurations change over time. This problem is usually addressed by augmenting the state vector to include the movement of dynamic entities, or by filtering the sensor data to remove the effects of environment dynamics.

16.1.3 Passive vs. Active

Information collected via measurements is crucial for robot localization. Therefore it is reasonable to consider localization problems where the robot can *explicitly* choose its actions to gather more (or more specific) information from the environment.

- *Passive localization* problems assume that the robot's motion is unrelated to its localization process.

- *Active localization* problems consider the ability of the robot to choose its actions (at least partially) to improve its understanding of the environment. For example, a robot in the corner of a room might choose to reorient itself to face the rest of the room, so it can collect environmental information as it moves along the wall. Hybrid approaches are also possible, since it may be inefficient to use active localization all of the time.

16.1.4 Single Robot vs. Multi-Robot

It is of course also possible to consider problems where several robots all gather independent information and then share that information with each other.

- *Single-robot localization* problems are the most commonly studied and utilized approach, and are often simpler because all data is collected on a single platform.

- *Multi-robot localization* problems consider teams of robots that share information in such a way that one robot's belief can be used to influence another robot's belief if the relative location between robots is known.

16.2 Robot Localization via Bayesian Filtering

The parametric (e.g. EKF) and non-parametric (e.g. particle) filters from the previous chapters are all variations of the Bayes filter. In particular they rely on a Markov process assumption and the identification of probabilistic measurement models. In this section it is shown how map-based robot localization can be cast into this framework, such that the previously discussed algorithms can be applied.

Similar to the general filtering context from the previous chapters, at time t the state is denoted by $\boldsymbol{x}_t$, the control input is denoted by $\boldsymbol{u}_t$, and the measurements are denoted by $\boldsymbol{z}_t$. For example, a differential drive robot equipped with a laser range-finder (returning a set of range measurements r_i and bearings ϕ_i), the state, control, and measurements would be:

$$\boldsymbol{x}_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad \boldsymbol{u}_t = \begin{bmatrix} v \\ \omega \end{bmatrix}, \quad \boldsymbol{z}_t = \begin{bmatrix} r_1 \\ \phi_1 \\ \vdots \end{bmatrix}. \tag{16.1}$$

However, the critical new component is the concept of a *map* (denoted as $\boldsymbol{m}$), which is a list of objects in the environment along with their properties:

$$\boldsymbol{m} = \{m_1, m_2, \dots, m_N\}, \tag{16.2}$$

where m_i represents the properties of a specific object. Generally there are two types of maps that will be considered, location-based maps and feature-based maps, which typically have differences in both computational efficiency and expressiveness.

For location-based maps, the index i associated with object m_i corresponds to a specific *location* (i.e. m_i are volumetric objects). For example, objects m_i in a location-based map might represent cells in a cell decomposition or grid representation of a map (see Figure 16.1). One potential disadvantage of the cell-based maps is that their resolution is dependent on the size of the cells, but their advantage is that they can explicitly encode information about presence (or absence) of objects in specific locations.

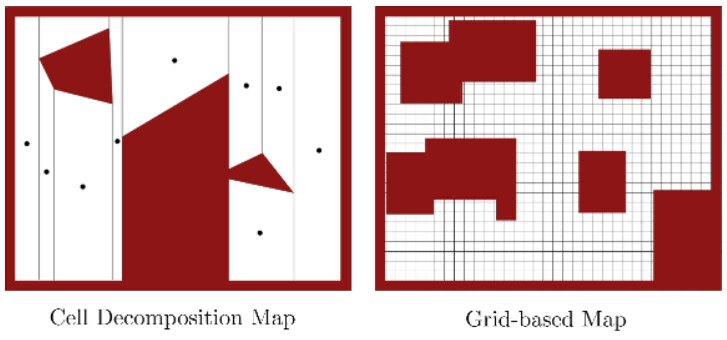

Figure 16.1: Two examples of location-based maps, both represent the map as a set of volumetric objects (i.e. cells in these cases).

For feature-based maps, an index i is a feature index, and m_i contains information about the properties of that feature, including its Cartesian location. These types of maps can typically be thought of as a collection of landmarks. Figure 16.2 gives two examples of feature-based maps, one which is represented by a set of lines, and another which is represented by nodes and edges like a graph (i.e. a topological map). Feature-based maps can be more finely tuned to specific environments, for example the line-based map might make sense to use in highly structured environments such as buildings. While feature-based maps can be computationally efficient, their main disadvantage is that they typically do not capture spatial information about all potential obstacles.

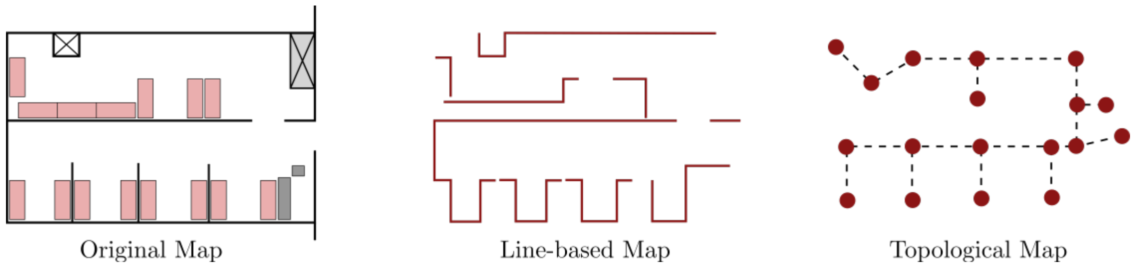

Figure 16.2: Two examples of feature-based maps.

16.2.1 *State Transition Model*

In the previous chapters on Bayesian filtering the probabilistic state transition model $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ describes the posterior distribution over the states that the robot could transition to when executing control $\boldsymbol{u}_t$ from $\boldsymbol{x}_{t-1}$. However in robot localization problems it might be important to take into account how the map $\boldsymbol{m}$ could affect the state transition since in general:

$$p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \neq p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}, \boldsymbol{m}).$$

For example, $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ cannot account for the fact that a robot cannot move through walls since it doesn't know that walls exist!

However, a common approximation is to make the assumption that:

$$p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}, \boldsymbol{m}) \approx \eta \frac{p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_t \mid \boldsymbol{m})}{p(\boldsymbol{x}_t)}, \tag{16.3}$$

where η is a normalization constant. This approximation can be derived from Bayes' rule by assuming that $p(\boldsymbol{m} \mid \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \approx p(\boldsymbol{m} \mid \boldsymbol{x}_t)$ (which is a tight approximation under high update rates). More specifically:

$$\begin{aligned}
p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}, \boldsymbol{m}) &= \frac{p(\boldsymbol{m}|\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}{p(\boldsymbol{m} \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}, \\
&= \eta' p(\boldsymbol{m}|\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \\
&\approx \eta' p(\boldsymbol{m}|\boldsymbol{x}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \\
&= \eta \frac{p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_t \mid \boldsymbol{m})}{p(\boldsymbol{x}_t)},
\end{aligned}$$

where η' and η are normalization constants (such that the total probability density integrates to one).

In this approximation the term $p(\boldsymbol{x}_t \mid \boldsymbol{m})$ is the state probability conditioned on the map which can be thought of as describing the "consistency" of state with respect to the map. The approximation (16.3) can therefore be viewed as making a probabilistic guess using the original state transition model (without map knowledge), and then using the consistency term $p(\boldsymbol{x}_t \mid \boldsymbol{m})$ to check the plausibility of the new state $\boldsymbol{x}_t$ given the map.

16.2.2 Measurement Model

The probabilistic measurement model model $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ from previous chapters also needs to be modified to take map information into account. This new measurement model can simply be expressed as $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{m})$ (i.e. measurement is also conditioned on the map). This is obviously important because the local measurements can have significant influence from the environment. For example a range measurement is dependent on what object is currently in the line of sight.

Additionally, since the suite of sensors on a robot may generate more than one measurement when queried, it is also common to make another measurement model assumption for simplicity. Suppose K measurements are taken at time t, such that:

$$\boldsymbol{z}_t = \begin{bmatrix} z_t^1 \\ \vdots \\ z_t^K \end{bmatrix}.$$

Then it can often be assumed that each of the K measurements are conditionally independent from each other (i.e. when conditioned on $\boldsymbol{x}_t$ and $\boldsymbol{m}$ the probability of measuring z_t^k is independent from the other measurements). With this assumption the probabilistic measurement model can be expressed as:

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{m}) = \prod_{k=1}^{K} p(z_t^k \mid \boldsymbol{x}_t, \boldsymbol{m}). \tag{16.4}$$

16.3 Markov Localization

With the probabilistic state transition and measurement models that include the map, the Bayes' filter can be directly modified as shown in Algorithm 9. As can

Algorithm 9: Markov Localization Algorithm

```
Data: $bel(\boldsymbol{x}_{t-1}), \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{m}$
Result: $bel(\boldsymbol{x}_t)$
foreach $\boldsymbol{x}_t$ do
    $\overline{bel}(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}, \boldsymbol{m}) bel(\boldsymbol{x}_{t-1}) d\boldsymbol{x}_{t-1}$
    $bel(\boldsymbol{x}_t) = \eta p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{m}) \overline{bel}(\boldsymbol{x}_t)$
return $bel(\boldsymbol{x}_t)$
```

be seen, this algorithm is conceptually identical to the Bayes' filter except for the inclusion of the model $\boldsymbol{m}$. This algorithm is referred to as the *Markov localization* algorithm, and the localization problem it is trying to solve is generally referred to as simply *Markov localization*[2].

The Markov localization algorithm can be used to address global localization, position tracking, and kidnapped robot problems, but generally some implementation details might be different. The choice for the initial (prior) belief

[2] Recall the use of the Markov property assumption in the derivation of the Bayes' filter.

distribution $bel(\boldsymbol{x}_0)$ is one such parameter that may be different depending on the type of localization problem.

Specifically, since the initial belief encodes any prior knowledge about the robot pose, the best choice of distribution depends on what (if any) knowledge is available. For example, in the position tracking problem it is assumed that an initial pose of the robot is known. Therefore choosing a (unimodal) Gaussian distribution $bel(\boldsymbol{x}_0) \sim \mathcal{N}(\bar{x}_0, \boldsymbol{\Sigma}_0)$ with a small covariance might be a good choice. Alternatively, for a global localization problem the initial pose is not known. In this case an appropriate choice for the initial belief would be a uniform distribution $bel(\boldsymbol{x}_0) = 1/|X|$ over all possible states $\boldsymbol{x}$.

Similarly to the original Bayes' filter from previous chapters, the Markov localization algorithm 9 is generally not possible to implement in a computationally tractable way. However, practical implementations can still be developed by again leveraging some sort of structure to the belief distribution $bel(\boldsymbol{x}_t)$ (e.g. through Gaussian or particle representations). Two commonly used implementations based on specific structured beliefs will now be discussed: extended Kalman filter localization and Monte Carlo localization.

16.4 Extended Kalman Filter (EKF) Localization

The extended Kalman filter (EKF) localization algorithm is essentially equivalent to the EKF algorithm presented in previous chapters, except that it also takes the map $\boldsymbol{m}$ into account. In particular, it still makes a Guassian belief assumption, $bel(\boldsymbol{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, to add structure to the filtering problem. As a brief review, the assumed state transition model is given by:

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

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_t)$ is Gaussian zero-mean noise. The Jacobian G_t is again defined by $G_t = \nabla_{\boldsymbol{x}} g(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$, where $\boldsymbol{\mu}_{t-1}$ is the expected value of the previous belief distribution $bel(\boldsymbol{x}_{t-1})$.

The main difference in EKF localization is the assumption that a feature-based map is available, consisting of point landmarks given by:

$$\boldsymbol{m} = \{m_1, m_2, \dots, m_N\}, \quad m_j = (m_{j,x}, m_{j,y}),$$

where N is the total number of landmarks, and each landmark m_j encapsulates the location $(m_{j,x}, m_{j,y})$ of the landmark in the global coordinate frame. Measurements $\boldsymbol{z}_t$ associated with these point landmarks at a time t are denoted by:

$$\boldsymbol{z}_t = \{z_t^1, z_t^2, \dots\},$$

where z_t^i is associated with a particular landmark and is assumed to be generated by the measurement model:

$$z_t^i = h(\boldsymbol{x}_t, j, \boldsymbol{m}) + \delta_t,$$

where $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ 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.

One fundamental problem that now needs to be addressed is the *data association problem*, which arises due to uncertainty in which measurements are associated with which landmark. To begin addressing this problem, the correspondences are modeled through a variable $c_t^i \in \{1, \dots, N+1\}$, which take on the values $c_t^i = j$ if measurement i corresponds to landmark j, and $c_t^i = N+1$ if measurement i has no corresponding landmark. Then, given a correspondence c_t^i of measurement i (associated with a specific landmark), the Jacobian H_t^i used in the EKF measurement correction step can be determined. Specifically, for the i-th measurement the Jacobian of the new measurement model can be computed by $H_t^{c_t^i} = \nabla_{\boldsymbol{x}} h(\bar{\boldsymbol{\mu}}_t, c_t^i, \boldsymbol{m})$, where $\bar{\boldsymbol{\mu}}_t$ is the predicted mean (that results from the EKF prediction step).

16.4.1 EKF Localization with Known Correspondences

In practice the correspondences between measurements $\boldsymbol{z}_t^i$ and landmarks m_j are generally unknown. However, it is useful from a pedagogical standpoint to first consider the case where these correspondences $\boldsymbol{c}_t = [c_t^1, \dots]^\top$ are assumed to be *known*.

In the EKF localization algorithm given in Algorithm 10, the main difference from the original EKF filter algorithm is that multiple measurements are processed at the same time. Crucially, this is accomplished in a computationally efficient way by exploiting the conditional independence assumption (16.4) for the measurements. In fact, by exploiting this assumption and some special properties of Gaussians, the multi-measurement update can be implemented by just looping over each measurement individually and applying the standard EKF correction.

Algorithm 10: Extended Kalman Filter Localization Algorithm

```
Data: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{c}_t, \boldsymbol{m}$
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 $z_t^i$ do
    $j = c_t^i$
    $S_t^i = H_t^j \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T + \boldsymbol{Q}_t$
    $K_t^i = \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T [S_t^i]^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_t^i (\boldsymbol{z}_t^i - h(\bar{\boldsymbol{\mu}}_t, j, \boldsymbol{m}))$
    $\bar{\boldsymbol{\Sigma}}_t = (I - K_t^i H_t^j) \bar{\boldsymbol{\Sigma}}_t$
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t$
return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
```

16.4.2 EKF Localization with Unknown Correspondences

For EKF localization with *unknown* correspondences, the correspondence variables must also be estimated! The simplest way to determine the correspondences online is to use maximum likelihood estimation, in which the most likely value of the correspondences $\boldsymbol{c}_t$ is determined by maximizing the data likelihood:

$$\hat{c}_t = \arg\max_{c_t} p(\boldsymbol{z}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{m}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t})$$

In other words, the set of correspondence variables is chosen to maximize the probability of getting the current measurement given the history of correspondence variables, the map, the history of measurements, and the history of controls. By marginalizing over the current pose $\boldsymbol{x}_t$ this distribution can be written as:

$$\begin{aligned} p(\boldsymbol{z}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{m}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) &= \int p(\boldsymbol{z}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{x}_t, \boldsymbol{m}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) p(\boldsymbol{x}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{m}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) d\boldsymbol{x}_t, \\ &= \int p(\boldsymbol{z}_t \mid \boldsymbol{c}_t, \boldsymbol{x}_t, \boldsymbol{m}) \overline{bel}(\boldsymbol{x}_t) d\boldsymbol{x}_t. \end{aligned}$$

Note that the term $p(\boldsymbol{z}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{x}_t, \boldsymbol{m})$ is essentially the assumed measurement model given *known* correspondences. Then, by again leveraging the conditional independence assumption for the measurements z_t^i from (16.4), this can be written as:

$$p(\boldsymbol{z}_t \mid \boldsymbol{c}_{1:t}, \boldsymbol{m}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) = \int \prod_i p(z_t^i \mid c_t^i, \boldsymbol{x}_t, \boldsymbol{m}) \overline{bel}(\boldsymbol{x}_t) d\boldsymbol{x}_t.$$

Importantly, each decision variable c_t^i in the maximization of this quantity shows up in separate terms of the product! Therefore it is possible to maximize each parameter independently by solving the optimization problems:

$$\hat{c}_t^i = \arg\max_{c_t^i} \int p(z_t^i \mid c_t^i, \boldsymbol{x}_t, \boldsymbol{m}) \overline{bel}(\boldsymbol{x}_t) d\boldsymbol{x}_t.$$

This problem can be solved quite efficiently since it is assumed that the measurement models and belief distributions are Gaussian[3]. In particular, the probability distribution resulting from the integral is a Gaussian with mean and covariance:

[3] Similar to the previous chapters, in this case the product of terms inside the integral will be Gaussian since both terms are Gaussian.

$$\int p(z_t^i \mid c_t^i, \boldsymbol{x}_t, \boldsymbol{m}) \overline{bel}(\boldsymbol{x}_t) d\boldsymbol{x}_t \sim \mathcal{N}(h(\bar{\boldsymbol{\mu}}_t, c_t^i, \boldsymbol{m}), H_t^{c_t^i} \bar{\boldsymbol{\Sigma}}_t [H_t^{c_t^i}]^\top + \boldsymbol{Q}_t).$$

The maximum likelihood optimization problem can therefore be expressed as:

$$\hat{c}_t^i = \arg\max_{c_t^i} \ \mathcal{N}(z_t^i \mid \hat{z}_t^{c_t^i}, \ S_t^{c_t^i}),$$

where $\hat{z}_t^j = h(\bar{\boldsymbol{\mu}}_t, j, \boldsymbol{m})$ and $S_t^j = H_t^j \bar{\boldsymbol{\Sigma}}_t [H_t^j]^\top + \boldsymbol{Q}_t$. To solve this maximization problem, recall the definition of the Gaussian distribution:

$$\mathcal{N}(z_t^i \mid \hat{z}_t^j, \ S_t^j) = \eta \exp\big(-\frac{1}{2}(z_t^i - \hat{z}_t^j)^\top [S_t^j]^{-1} (z_t^i - \hat{z}_t^j) \big),$$

where η is a normalization constant. Since the exponential function is monotonically increasing and since η is a positive constant, the maximum likelihood estimation problem can be equivalently expressed as:

$$\hat{c}_t^i = \arg\min_{c_t^i} \; d_t^{i,c_t^i}, \tag{16.5}$$

where

$$d_t^{ij} = (z_t^i - \hat{z}_t^j)^\top [S_t^j]^{-1} (z_t^i - \hat{z}_t^j), \tag{16.6}$$

is referred to as the *Mahalanobis distance*.

The EKF localization algorithm with unknown correspondences is very similar to Algorithm 10, except with the addition of this maximum likelihood estimation step. This new algorithm is given in Algorithm 11.

Algorithm 11: EKF Localization Algorithm, Unknown Correspondences

```
**Data:** $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{m}$
**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** $z_t^i$ **do**
    **foreach** *landmark k in the map* **do**
        $\hat{z}_t^k = h(\bar{\boldsymbol{\mu}}_t, k, \boldsymbol{m})$
        $S_t^k = H_t^k \bar{\boldsymbol{\Sigma}}_t [H_t^k]^T + \boldsymbol{Q}_t$
    $j = \arg\min_k \; (z_t^i - \hat{z}_t^k)^\top [S_t^k]^{-1} (z_t^i - \hat{z}_t^k)$
    $K_t^i = \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T [S_t^j]^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_t^i (z_t^i - \hat{z}_t^j)$
    $\bar{\boldsymbol{\Sigma}}_t = (I - K_t^i H_t^j) \bar{\boldsymbol{\Sigma}}_t$
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t$
**return** $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
```

One of the disadvantages of using the maximum likelihood estimation is that it can be brittle with respect to outliers and in cases where there are equally likely hypothesis for the correspondence. An alternative approach to estimating correspondences that is more robust to outliers is to use a *validation gate*. In this approach the Mahalanobis smallest distance d_t^{ij} must also pass a *thresholding* test:

$$(z_t^i - \hat{z}_t^j)^\top [S_t^j]^{-1} (z_t^i - \hat{z}_t^j) \leq \gamma,$$

in order for a correspondence to be created.

Example 16.4.1 (Differential Drive Robot with Range and Bearing Measurements)**.** Consider a differential drive robot with state $\boldsymbol{x} = [x, y, \theta]^\top$, 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:

$$z_t = \{[r_t^1, \phi_t^1]^\top, [r_t^2, \phi_t^2]^\top, \dots\},$$

where each measurement z_t^i contains the range r_t^i and bearing ϕ_t^i.

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

$$h(\mathbf{x}_t, j, \mathbf{m}) = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{bmatrix}. \tag{16.7}$$

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

$$H_t^j = \begin{bmatrix} -\frac{m_{j,x} - \bar{\mu}_{t,x}}{\sqrt{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2}} & -\frac{m_{j,y} - \bar{\mu}_{t,y}}{\sqrt{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2}} & 0 \\ \frac{m_{j,y} - \bar{\mu}_{t,y}}{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2} & -\frac{m_{j,x} - \bar{\mu}_{t,x}}{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2} & -1 \end{bmatrix}. \tag{16.8}$$

It is also common to assume that the covariance of the measurement noise is given by:

$$\mathbf{Q}_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix},$$

where σ_r is the standard deviation of the range measurement noise and σ_ϕ is the standard deviation of the bearing measurement noise. This diagonal covariance matrix is typically used since these two measurements can be assumed to be uncorrelated.

16.5 *Monte Carlo Localization (MCL)*

Another approach to Markov localization is the Monte Carlo localization (MCL) algorithm. This algorithm leverages the non-parametric particle filter algorithm from the previous chapter, and is therefore much better suited to solving *global* localization problems (unlike EKF localization which only solves position tracking problems). MCL can also be used to solve the kidnapped robot problem through some small modifications, such as injecting new random particles at each step to ensure that a "particle collapse" problem does not occur.

As a brief review, the particle filter represents the belief $bel(\mathbf{x}_t)$ by a set of M particles:

$$\mathcal{X}_t := \{\mathbf{x}_t^{[1]}, \mathbf{x}_t^{[2]}, ..., \mathbf{x}_t^{[M]}\},$$

where each particle $\mathbf{x}_t^{[m]}$ represents a hypothesis about the true state $\mathbf{x}_t$. At each step of the algorithm the state transition model is used to propagate forward the particles, and then the measurement model is used to resample particles based on the measurement likelihood. This algorithm is shown in Algorithm 12, and is nearly identical to the particle filter algorithm except that the map $\mathbf{m}$ is used in the probabilistic state transition and measurement models.

Algorithm 12: Monte Carlo Localization Algorithm

```
Data: $\mathcal{X}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{m}$
Result: $\mathcal{X}_t$
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to $M$ do
    Sample $\bar{\boldsymbol{x}}_t^{[m]} \sim p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}^{[m]}, \boldsymbol{m})$
    $w_t^{[m]} = p(\boldsymbol{z}_t \mid \bar{\boldsymbol{x}}_t^{[m]}, \boldsymbol{m})$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t \cup \left(\bar{\boldsymbol{x}}_t^{[m]}, w_t^{[m]}\right)$
for $m = 1$ to $M$ do
    Draw $i$ with probability $\propto w_t^{[i]}$
    Add $\bar{\boldsymbol{x}}_t^{[i]}$ to $\mathcal{X}_t$
return $\mathcal{X}_t$
```

17
Simultaneous Localization and Mapping (SLAM)

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)[1].

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005

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 17.1. 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.

17.1 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

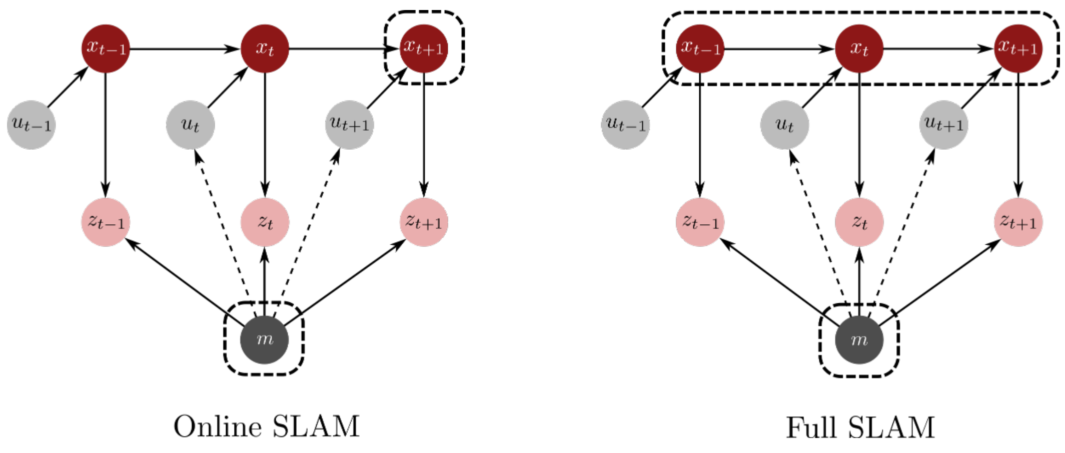

Figure 17.1: Difference between online and full SLAM, where online SLAM only estimates the current robot pose while full SLAM also estimates the robot's history.

distribution $bel(\boldsymbol{x}_t)$, and state transition and measurement models. It will also be assumed that the map is feature-based:

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

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:

$$\boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{m}_1 \\ \vdots \\ \boldsymbol{m}_N \end{bmatrix}, \tag{17.1}$$

where $\boldsymbol{m}_i = [m_{i,x}, m_{i,y}]^\top$. 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.

17.1.1 *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:

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

where $\delta_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t)$ 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 $\overline{bel}(\boldsymbol{y}_t)$.

17.1.2 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 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 $\boldsymbol{c}_t = [c_t^1, \dots]^\top$ are *known*.

Algorithm 13 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:

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

Algorithm 13: Extended Kalman Filter Online SLAM Algorithm

```
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 $z_t^i$ 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 $z_t^i$
    $S_t^i = H_t^j \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T + \boldsymbol{Q}_t$
    $K_t^i = \bar{\boldsymbol{\Sigma}}_t [H_t^j]^T [S_t^i]^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_t^i (z_t^i - h(\bar{\boldsymbol{\mu}}_t, j))$
    $\bar{\boldsymbol{\Sigma}}_t = (I - K_t^i H_t^j) \bar{\boldsymbol{\Sigma}}_t$
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t$
return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
```

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.

17.2 *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 14.

As can be seen there are a couple differences between Algorithm 13 and Algorithm 14. First, the measurements 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.

Algorithm 14: EKF Online SLAM Algorithm, Unknown Correspondences

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

In particular, extraneous 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 17.2.1 (Differential Drive Robot with Range and Bearing Measurements)**.** Consider a differential drive robot with state $x = [x, y, \theta]^\top$, and suppose a sensor is available on the robot which measures the range r and bearing ϕ of landmarks $m_j \in m$ relative to the robot's local coordinate frame. Additionally, multiple measurements corresponding to different features can be collected at each time step:

$$z_t = \{[r_t^1, \phi_t^1]^\top, [r_t^2, \phi_t^2]^\top, \dots\},$$

where each measurement 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:

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

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

$$h(\boldsymbol{y}_t, j) = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{bmatrix}. \tag{17.2}$$

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

$$H_t^j = \begin{bmatrix} -\frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_{t,j}}} & -\frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_{t,j}}} & 0 & 0 & \dots & 0 & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_{t,j}}} & \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_{t,j}}} & 0 & \dots \\ \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_{t,j}} & -\frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_{t,j}} & -1 & 0 & \dots & 0 & -\frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_{t,j}} & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_{t,j}} & 0 & \dots \end{bmatrix}, \tag{17.3}$$

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 13) to initialize the landmark position and can be used in the unknown-correspondence case (Algorithm 14) to hypothesize the position of new landmarks.

17.3 *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 (17.1).

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.

17.3.1 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:

$$p(\boldsymbol{y}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) = p(\boldsymbol{x}_{1:t} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}, c_{1:t}), \tag{17.4}$$

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 (17.4) 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:

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

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

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

17.3.2 Fast SLAM with Known Correspondences

The particle SLAM algorithm referred to as Fast SLAM uses the factorization of the posterior $p(\boldsymbol{y}_{1:t} \mid z_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ in (17.4) to decompose the full SLAM problem into more manageable sub-problems. Specifically, the path posterior $p(\boldsymbol{x}_{1:t} \mid z_{1:t}, \boldsymbol{u}_{1:t}, c_{1:t})$ is estimated using a particle filter and the feature posteriors $p(\boldsymbol{m}_n \mid \boldsymbol{x}_{1:t}, z_{1:t}, c_{1:t})$ 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 15.

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.

Algorithm 15: Fast SLAM Algorithm

```
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
        $\hat{\boldsymbol{z}}^{[k]} = h(\boldsymbol{\mu}_{j,t-1}^{[k]}, \boldsymbol{x}_t^{[k]})$
        $S = H^j \boldsymbol{\Sigma}_{j,t-1}^{[k]} [H^j]^T + \boldsymbol{Q}_t$
        $K = \boldsymbol{\Sigma}_{j,t-1}^{[k]} [H^j]^T [S]^{-1}$
        $\boldsymbol{\mu}_{j,t}^{[k]} = \boldsymbol{\mu}_{j,t-1}^{[k]} + K(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})$
        $\boldsymbol{\Sigma}_{j,t}^{[k]} = (I - KH^j)\boldsymbol{\Sigma}_{j,t-1}^{[k]}$
        $w^{[k]} = \det(2\pi S)^{-1/2} \exp\big(-\frac{1}{2}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})\boldsymbol{Q}^{-1}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})\big)$
    for $n \in \{1, \dots, N\}$, $n \neq c_t$ do
        $\boldsymbol{\mu}_{n,t}^{[k]} = \boldsymbol{\mu}_{n,t-1}^{[k]}$
        $\boldsymbol{\Sigma}_{n,t}^{[k]} = \boldsymbol{\Sigma}_{n,t-1}^{[k]}$
$\mathcal{Y}_t = \emptyset$
for $m = 1$ to $M$ do
    Draw $k$ with probability $\propto w_t^{[k]}$
    $\mathcal{Y}_t = \mathcal{Y}_t \cup (\bar{\boldsymbol{x}}_t^{[k]}, \boldsymbol{\mu}_{1,t}^{[k]}, \boldsymbol{\Sigma}_{1,t}^{[k]}, \dots, \boldsymbol{\mu}_{N,t}^{[k]}, \boldsymbol{\Sigma}_{N,t}^{[k]})$
return $\mathcal{Y}_t$
```

17.4 *Exercises*

17.4.1 *EKF SLAM*

Complete *Problem 2: EKF SLAM* located in the online repository:

`https://github.com/PrinciplesofRobotAutonomy/AA274A_HW4,`

where you will implement an EKF SLAM algorithm. Note that the EKF localization exercise from the chapter on parametric filters should be completed first.