7

MOBILE ROBOT LOCALIZATION

7.1 INTRODUCTION

This chapter presents a number of probabilistic algorithms for mobile robot localization. Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment. It is often called *position estimation* or *position tracking*. Mobile robot localization is an instance of the general localization problem, which is the most basic perceptual problem in robotics. This is because nearly all robotics tasks require knowledge of the location of the robots and the objects that are being manipulated (although not necessarily within a global map).

Localization can be seen as a problem of coordinate transformation. Maps are described in a global coordinate system, which is independent of a robot's pose. Localization is the process of establishing correspondence between the map coordinate system and the robot's local coordinate system. Knowing this coordinate transformation enables the robot to express the location of objects of interests within its own coordinate frame—a necessary prerequisite for robot navigation. As the reader easily verifies, knowing the pose $x_t = (x \;\; y \;\; \theta)^T$ of the robot is sufficient to determine this coordinate transformation, assuming that the pose is expressed in the same coordinate frame as the map.

Unfortunately—and herein lies the problem of mobile robot localization—the pose can usually *not* be sensed directly. Put differently, most robots do not possess a (noise-free!) sensor for measuring pose. The pose has therefore to be inferred from data. A key difficulty arises from the fact that a single sensor measurement is usually insufficient to determine the pose. Instead, the robot has to integrate data over time to determine its pose. To see why this is necessary, just picture a robot located inside a

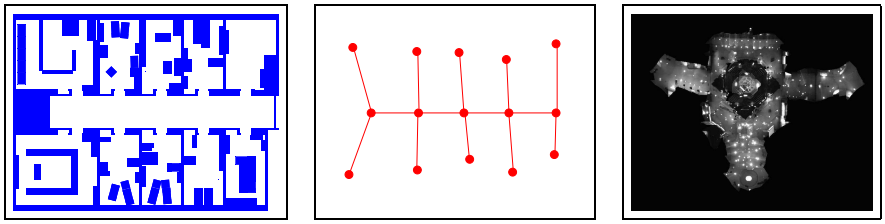

Figure 7.1 Example maps used for robot localization: a 2D metric layout, a graph-like topological map, and an image mosaic of a ceiling

building where many corridors look alike. Here a single sensor measurement (e.g., a range scan) is usually insufficient to disambiguate the identity of the corridor.

Localization has been applied in conjunction with a broad set of map representations. We already discussed two types of maps in the previous chapter: feature-based and location-based. An example of the latter were occupancy grid maps, which were informally discussed and are subject to a later chapter in this book. Instances of such maps are shown in Figure 7.1. This figure shows a hand-drawn metric 2-D map, a graph-like topological map, and an image mosaic of a ceiling (which can also be used as a map). The space of map representations used in today's research is huge. A number of later chapters will investigate specific map types and discuss algorithms for acquiring maps from data. Localization assumes that an accurate map is available.

In this and the subsequent chapter, we will present some basic probabilistic algorithms for mobile localization. All of these algorithms are variants of the basic Bayes filter described in Chapter 2. We will discuss the advantages and shortcomings of each representation and associated algorithm. The chapter also goes through a series of extensions that address different localization problems, as defined through the following taxonomy of robot localization problems.

7.2 A TAXONOMY OF LOCALIZATION PROBLEMS

Not every localization problem is equally hard. To understand the difficulty of a localization problem, we will now discuss a brief taxonomy of localization problems. This taxonomy will divide localization problems along a number of important dimensions

pertaining to the nature of the environment and the initial knowledge that a robot may possess relative to the localization problem.

Local Versus Global Localization

Localization problems are characterized by the type of knowledge that is available initially and at run-time. We distinguish three types of localization problems with an increasing degree of difficulty.

- **Position tracking.** Position tracking assumes that the *initial* robot pose is known. Localizing the robot can be achieved by accommodating the noise in robot motion. The effect of such noise is usually small. Hence, methods for position tracking often rely on the assumption that the pose error is small. The pose uncertainty is often approximated by a unimodal distribution (e.g., a Gaussian). The position tracking problem is a *local* problem, since the uncertainty is local and confined to region near the robot's true pose.
- **Global localization.** Here the initial pose of the robot is unknown. The robot is initially placed somewhere in its environment, but it lacks knowledge of where it is. Approaches to global localization cannot assume boundedness of the pose error. As we shall see later in this chapter, unimodal probability distributions are usually inappropriate. Global localization is more difficult than position tracking; in fact, it subsumes the position tracking problem.
- **Kidnapped robot problem.** This problem is a variant of the global localization problem, but one that is even more difficult. During operation, the robot can get kidnapped and teleported to some other location. The kidnapped robot problem is more difficult than the global localization problem, in that the robot might believe it knows where it is while it does not. In global localization, there robots knows that it doesn't know where it is. One might argue that robots are rarely kidnapped in practice. The practical importance of this problem, however, arises from the observation that most state-of-the-art localization algorithms cannot be guaranteed never to fail. The ability to recover from failures is essential for truly autonomous robots. Testing a localization algorithm by kidnapping it measures its ability to recover from global localization failures.

Static Versus Dynamic Environments

A second dimension that has a substantial impact on the difficulty of localization is the environment. Environments can be static or dynamic.

- **Static environments.** Static environments are environments where the only variable quantity (state) is the robot's pose. Put differently, only the robot moves in static environment. All other objects in the environments remain at the same location forever. Static environments have some nice mathematical properties that make them amenable to efficient probabilistic estimation.

- **Dynamic environments.** Dynamic environments possess objects other than the robot whose location or configuration changes over time. Of particular interest are changes that persist over time, and that impact more than a single sensor reading. Changes that are not measurable are of course of no relevance to localization, and those that affect only a single measurement are best treated as noise (*cf.* Chapter 2.4.4). Examples of more persistent changes are: people, daylight (for robots equipped with cameras), movable furniture, or doors. Clearly, most real environment are dynamic, with state changes occurring at a range of different speeds.

Obviously, localization in dynamic environments is more difficult than localization in static ones. There are two principal approaches for accommodating dynamics: First, dynamic entities might be included in the state vector. As a result, the Markov assumption might now be justified, but such an approach carries the burden of additional computational and modeling complexity; in fact, the resulting algorithm becomes effectively a mapping algorithm. Second, in certain situations sensor data can be filtered so as to eliminate the damaging effect of unmodeled dynamics. Such an approach will be described further below in Section 8.4.

Passive Versus Active Approaches

A third dimension that characterizes different localization problems pertains to the fact whether or not the localization algorithm controls the motion of the robot. We distinguish two cases:

- **Passive localization.** In passive approaches, the localization module only *observes* the robot operating. The robot is controlled through some other means, and the robot's motion is not aimed at facilitating localization. For example, the robot might move randomly or perform its everyday's tasks.

- **Active localization.** Active localization algorithms control the robot so as to minimize the localization error and/or the costs arising from moving a poorly localized robot into a hazardous place.

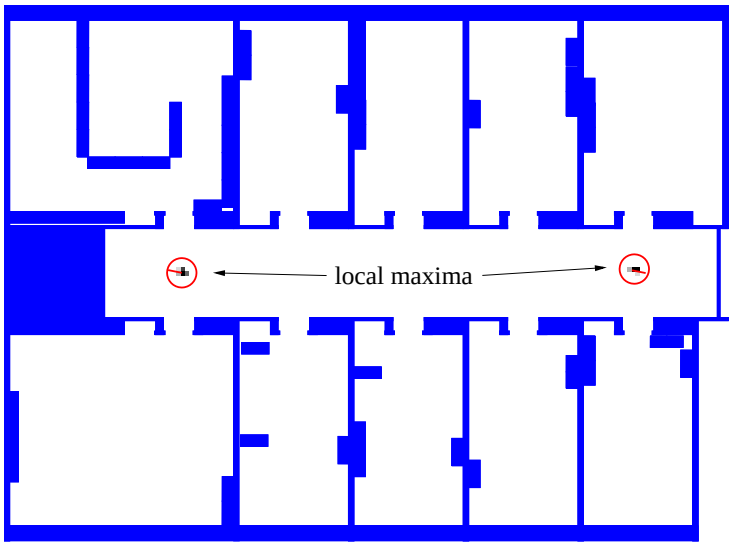

Figure 7.2 Example situation that shows a typical belief state during global localization in a locally symmetric environment. The robot has to move into one of the rooms to determine its location.

Active approaches to localization typically yield better localization results than passive ones. We already discussed an examples in the introduction to this book: coastal navigation. A second example situation is shown in Figure 7.2. Here the robot is located in a symmetric corridor, and its belief after navigating the corridor for a while is centered at two (symmetric) poses. The local symmetry of the environment makes it impossible to localize the robot while in the corridor. Only if it moves into a room will it be able to eliminate the ambiguity and to determine its pose. It is situations like these where active localization gives much better results: Instead of merely waiting until the robot incidentally moves into a room, active localization can recognize the impasse and send it there directly.

However, a key limitation of active approaches is that they require control over the robot. Thus, in practice, an active localization technique alone tends to be insufficient: The robot has to be able to localize itself even when carrying out some other task than localization. Some active localization techniques are built on top of a passive technique. Others combine tasks performance goals with localization goals when controlling a robot.

This chapter exclusively considers passive localization algorithms. We will return to the issue of active localization in Chapter **??** of this book, where we will present probabilistic algorithms for robot control.

Single-Robot Versus Multi-Robot

A fourth dimension of the localization problem is related to the number of robots involved.

- **Single-robot localization.** The most commonly studied approach to localization deals with a single robot only. Single robot localization offers the convenience that all data is collected at a single robot platform, and there is no communication issue.
- **Multi-robot localization.** The localization problem naturally arises to teams of robots. At first glance, each robot could localize itself individually, hence the multi-robot localization problem can be solved through single-robot localization. If robots are able to detect each other, however, there is the opportunity to do better. This is because one robot's belief can be used to bias another robot's belief if knowledge of the relative location of both robots is available. The issue of multi-robot localization raises interesting, non-trivial issues on the representation of beliefs and the nature of the communication between them.

These four dimensions capture the four most important characteristics of the mobile robot localization problem. There exist a number of other characterizations that impact the hardness of the problem, such as the information provided by robot measurements and the information lost through motion. Also, symmetric environments are more difficult than asymmetric ones, due to the higher degree of ambiguity. We will now look at specific algorithms and discuss their applicability to the different localization problems as defined thus far.

7.3 MARKOV LOCALIZATION

Probabilistic localization algorithms are variants of the Bayes filter. The straightforward application of Bayes filters to the localization problem is called *Markov localization*. Table 7.1 depicts the basic algorithm. This algorithm is derived from the algorithm **Bayes_filter** (Table 2.1 on page 24). Notice that **Markov_localization** also requires a map m as input. The map plays a role in the measurement model $p(z_t \mid x_t, m)$ (Line 4). It often, but not always, is incorporated in the motion model

1: **Algorithm Markov_localization(**$bel(x_{t-1}), u_t, z_t, m$**):**
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

Table 7.1 Markov localization.

$p(x_t \mid u_t, x_{t-1}, m)$ as well (Line 3). Just like the Bayes filter, Markov localization transforms a probabilistic belief at time $t-1$ into a belief at time t. Markov localization addresses the global localization problem, the position tracking problem, and the kidnapped robot problem in static environments.

The initial belief, $bel(x_0)$, reflects the initial knowledge of the robot's pose. It is set differently depending on the type of localization problem.

- **Position tracking.** If the initial pose is known, $bel(x_0)$ is initialized by a point-mass distribution. Let $\bar{x}_0$ denote the (known) initial pose. Then

$$bel(x_0) = \begin{cases} 1 & \text{if } x_0 = \bar{x}_0 \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

  Point-mass distributions are discrete and therefore do not possess a density.

  In practice the initial pose is often just known in approximation. The belief $bel(x_0)$ is then usually initialized by a narrow Gaussian distribution centered around $\bar{x}_0$. Gaussians were defined in Equation (2.4) on page 11.

$$\begin{aligned} bel(x_0) &= \det(2\pi\Sigma)^{-\frac{1}{2}}\ \exp\left\{-\tfrac{1}{2}(x_0-\bar{x}_0)^T\Sigma^{-1}(x_0-\bar{x}_0)\right\} \\ &\sim \mathcal{N}(x_0; \bar{x}_0, \Sigma) \end{aligned} \tag{7.2}$$

  Σ is the covariance of the initial pose uncertainty.

- **Global localization**. If the initial pose is unknown, $bel(x_0)$ is initialized by a uniform distribution over the space of all legal poses in the map:

$$bel(x_0) = \frac{1}{|X|} \tag{7.3}$$

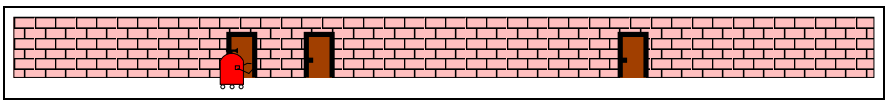

Figure 7.3 Example environment used to illustrate mobile robot localization: One-dimensional hallway environment with three indistinguishable doors. Initially the robot does not know its location except for its heading direction. Its goal is to find out where it is.

where $|X|$ stands for the volume (Lebesgue measure) of the space of all poses within the map.

- **Other.** Partial knowledge of the robot's position can usually easily be transformed into an appropriate initial distribution. For example, if the robot is known to start next to a door, one might initialize $bel(x_0)$ using a density that is zero except for places near doors, where it may be uniform. If it is known to be located in a specific corridor, one might initialize $bel(x_0)$ by a uniform distribution in the area of the corridor and zero anywhere else.

7.4 ILLUSTRATION OF MARKOV LOCALIZATION

We have already discussed Markov localization in the introduction to this book, as a motivating example for probabilistic robotics. Now we can back up this example using a concrete mathematical framework. Figure 7.3 depicts our one-dimensional hallway with three identically looking doors. The initial belief $bel(x_0)$ is uniform over all poses, as illustrated by the uniform density in Figure 7.4a. As the robot queries its sensors and notices that it is adjacent to one of the doors, it multiplies its belief $bel(x_0)$ by $p(z_t \mid x_t, m)$, as stated in Line 4 of our algorithm. The upper density in Figure 7.4b visualizes $p(z_t \mid x_t, m)$ for the hallway example. The lower density is the result of multiplying this density into the robot's uniform prior belief. Again, the resulting belief is multi-modal, reflecting the residual uncertainty of the robot at this point.

As the robot moves to the right, indicated in Figure 7.4c, Line 3 of the Markov locations algorithm convolves its belief with the motion model $p(x_t \mid u_t, x_{t-1})$. The motion model $p(x_t \mid u_t, x_{t-1})$ is not focused on a single pose but on a whole continuum of poses centered around the expected outcome of a noise-free motion. The effect is visualized in Figure 7.4c, which shows a shifted belief that is also flattened out, as a result of the convolution.

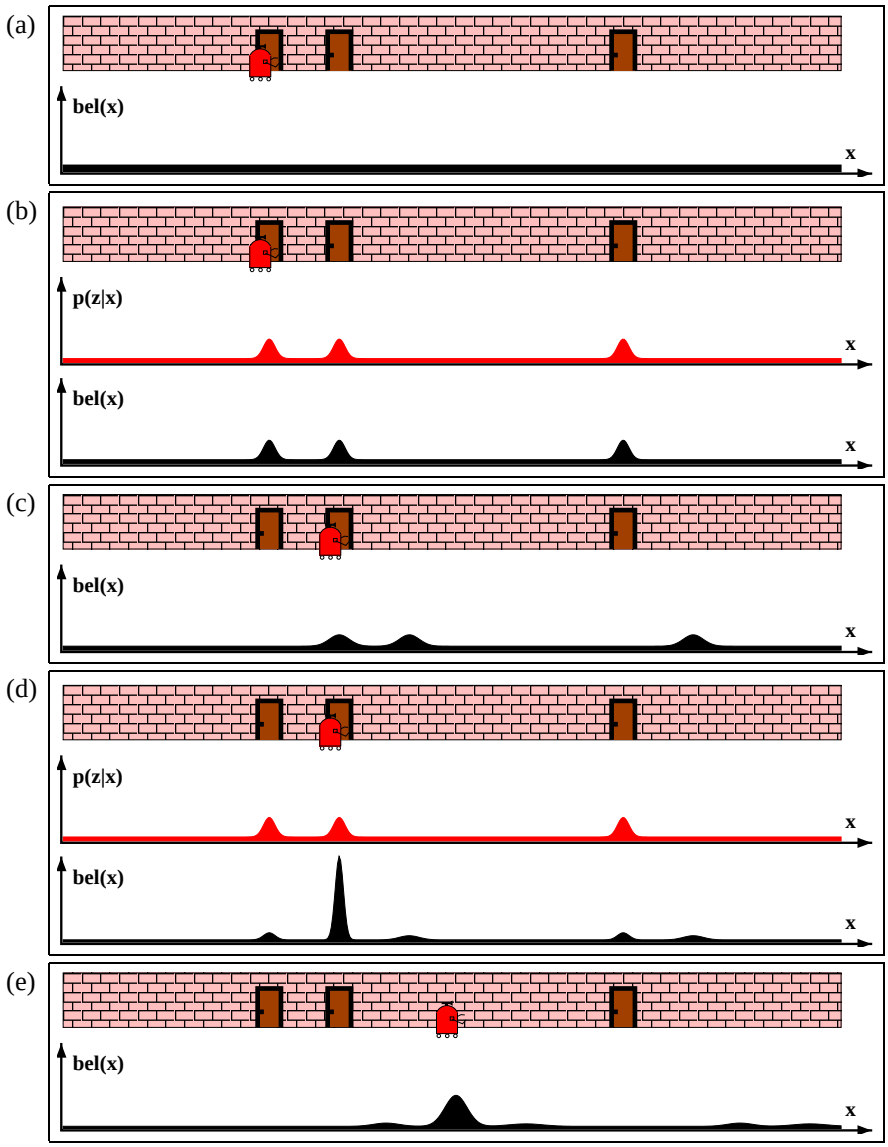

Figure 7.4 Illustration of the Markov localization algorithm. Each picture depicts the position of the robot in the hallway and its current belief $bel(x)$. (b) and (d) additionally depict the observation model $p(z_t \mid x_t)$, which describes the probability of observing a door at the different locations in the hallway.

The final measurement is illustrated in Figure 7.4d. Here the Markov localization algorithm multiplies the current belief with the perceptual probability $p(z_t \mid x_t)$. At this point, most of the probability mass is focused on the correct pose, and the robot is quite confident of having localized itself. Figure 7.4e illustrates the robot's belief after having moved further down the hallway.

We already noted that Markov localization is independent of the underlying representation of the state space. In fact, Markov localization can be implemented using any of the representations discussed in Chapter 2. We will now consider three different representations and devise practical algorithms that can localize mobile robots in real time. We begin with Kalman filters, which represent beliefs by their first and second moment. We then continue with discrete, grid representations and finally introduce algorithms using particle filters.

7.5 EKF LOCALIZATION

The extended Kalman filter localization algorithm, or EKF localization, is a special case of Markov localization. EKF localization represent beliefs $bel(x_t)$ by their their first and second moment, that is, the mean μ_t and the covariance Σ_t. The basic EKF algorithm was stated in Table 3.3 in Chapter 3.3. EKF localization shall be our first concrete implementation of an EKF in the context of an actual robotics problem.

Our EKF localization algorithm assumes that the map is represented by a collection of features. Thus, at any point in time t, the robot gets to observe a vector of ranges and bearings to nearby features: $z_t = \{z_t^1, z_t^2, \ldots\}$. We begin with a localization algorithm in which all features are uniquely identifiable. The existence of uniquely identifiable features may not be a bad assumption: For example, the Eiffel Tour in Paris is a landmark that is rarely confused with other landmarks, and it is widely visible throughout Paris. The identity of a feature is expressed by set of *correspondence variables*, denoted c_t^i, one for each feature vector z_t^i. Correspondence variables were already discussed in Chapter 6.6. In our first algorithm, the correspondence is assumed to be known. We then progress to a more general version which allows for ambiguity among features. Instead of the correspondence, the robot observes a feature signature, s_t^i. The second, more general version applied a maximum likelihood estimator to estimate the value of the latent correspondence variable, and uses the result of this estimation as ground truth.

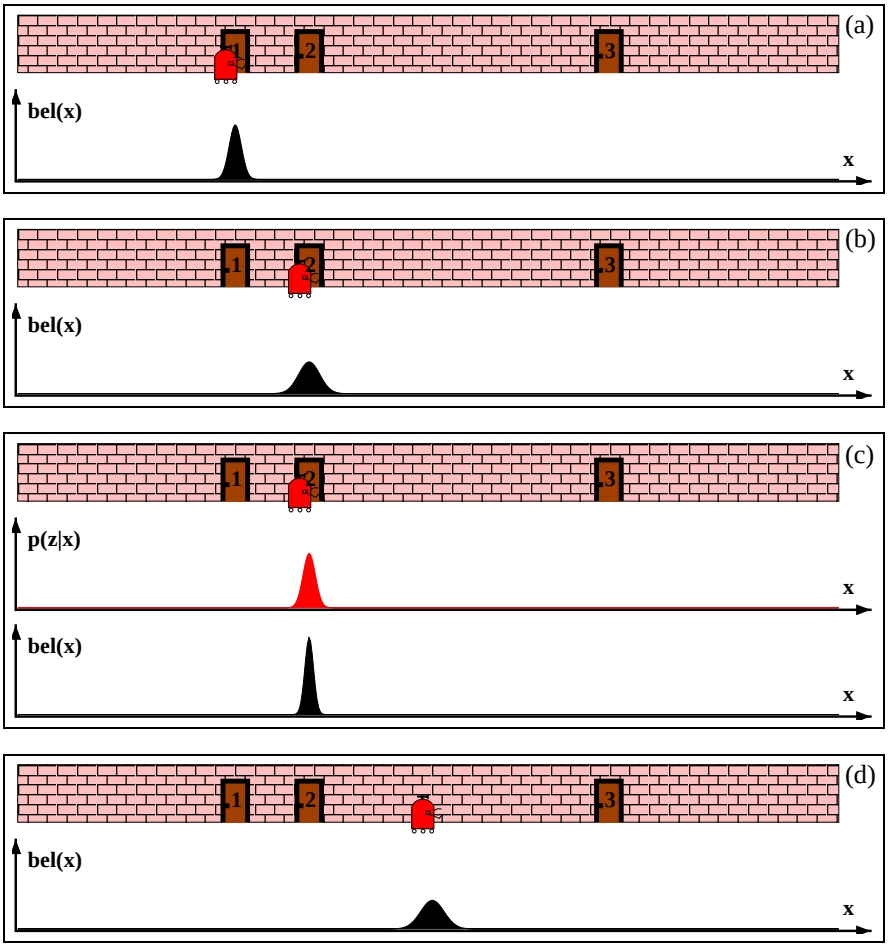

Figure 7.5 Application of the Kalman filter algorithm to mobile robot localization. All densities are represented by uni-modal Gaussians.

7.5.1 Illustration

Figure 7.5 illustrates the EKF localization algorithm using our example of mobile robot localization in the one-dimensional corridor environment (*cf.* Figure 7.3). To accommodate the unimodal shape of the belief in EKFs, we make two convenient assumptions: First, we assume that the correspondences are known: we will attach unique labels to each door (1, 2, and 3), and we will denote the measurement model by $p(z_t \mid x_t, c_t)$ where $c_t \in \{1, 2, 3\}$ is the identity of the door observed at time

t. Second, we assume that the initial pose is relatively well known. A typical initial belief is represented by the Gaussian distribution shown in Figure 7.5a, centered on the area near Door 1 and with a Gaussian uncertainty as indicated in that figure. As the robot moves to the right, its belief is convolved with the Gaussian motion model. The resulting belief is a shifted Gaussian of increased width, as shown in Figure 7.5b.

Now suppose the robot detects that it is in front of door $c_t = 2$. The upper density in Figure 7.5c visualizes $p(z_t \mid x_t, m, c_t)$ for this observation—again a Gaussian. Folding this measurement probability into the robot's belief yields the posterior shown in Figure 7.5c. Note that the variance of the resulting belief is smaller than the variances of both the robot's previous belief and the observation density. This is natural, since integrating two independent estimates should make the robot more certain than each estimate in isolation. After moving down the hallway, the robot's uncertainty in its position increases again, since the EKF continues to incorporate motion uncertainty into the robot's belief. Figure 7.5d shows one of these beliefs. This example illustrates the EKF in our limited setting.

7.5.2 The EKF Localization Algorithm

The discussion thus far has been fairly abstract: We have silently assumed the availability of an appropriate motion and measurement model, and have left unspecified a number of key variables in the EKF update. We will now discuss a concrete implementation of the EKF, for feature-based maps. Our feature-based maps consist of point landmarks, as already discussed in Chapter 6.2. For such point landmarks, we will use the common measurement model discussed in Chapter 6.6. We will also adopt the velocity motion model defined in Chapter 5.3. The reader may take a moment to briefly reacquire the basic measurement and motion equations discussed in these chapters before reading on.

Table 7.2 describes **EKF_localization_known_correspondences**, the EKF algorithm for localization with known correspondences. This algorithm is derived from the EKF in Table 3.3 in Chapter 3. It requires as its input a Gaussian estimate of the robot pose at time $t-1$, with mean μ_{t-1} and covariance Σ_{t-1}. Further, it requires a control u_t, a map m, and a set of features $z_t = \{z_t^1, z_t^2, \ldots\}$ measured at time t, along with the correspondence variables $c_t = \{c_t^1, c_t^2, \ldots\}$. It output is a new, revised estimate μ_t, Σ_t.

The individual calculations in this algorithm will be explained further below. Lines 2 through 4 implement the familiar motion update, using a linearized motion model. The predicted pose after motion is calculated as $\bar{\mu}_t$ in Line 2, and Line 4 computes the

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

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

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

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

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

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

7: $j = c_t^i$

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

9: $q = \delta^T\delta$

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

11: $H_t^i = \frac{1}{q}\begin{pmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 \\ \delta_y & \delta_x & -1 \\ 0 & 0 & 0 \end{pmatrix}$

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

13: *endfor*

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

15: $\Sigma_t = (I - \sum_i K_t^i\ H_t^i)\ \bar{\Sigma}_t$

16: *return* μ_t, Σ_t

Table 7.2 The extended Kalman filter (EKF) localization algorithm, formulated here for a feature-based map and a robot equipped with sensors for measuring range and bearing. This version assumes knowledge of the exact correspondences.

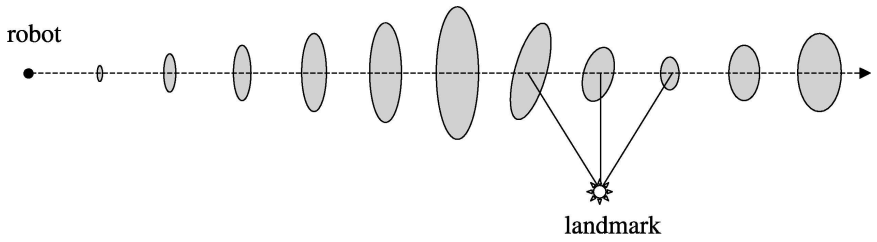

Figure 7.6 Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.

corresponding uncertainty ellipse. Lines 5 to 15 implement the measurement update. The core of this update is a loop through all possible features i observed at time t. In Line 7, the algorithm assigns to j the correspondence of the i-th feature in the measurement vector. It then calculates a predicted measurement $\hat{z}_t^i$ and the Jacobian H_t^i of the measurement model. The Kalman gain K_t^i is then calculated in Line 12 for each observed feature. The sum of all updates is then applied to obtain the new pose estimate, as stated in Lines 14 and 15. Notice that the last row of H_t^i is all zero. This is because the signature does not depend on the robot pose. The effect of this degeneracy is that the observed signature s_t^i has no effect on the result of the EKF update. This should come at no surprise: knowledge of the correct correspondence z_t^i renders the observed signature entirely uninformative.

Figure 7.6 illustrates the EKF localization algorithm in a synthetic environment with a single landmark. The robot starts out on the left and accrues uncertainty as it moves. Upon seeing the landmark, its uncertainty is gradually reduced as indicated.

7.5.3 Mathematical Derivation

To understand the motion update, let us briefly restate the motion model that was defined in Equation (5.13):

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t\Delta t) \\ \omega_t\Delta t + \gamma_t\Delta t \end{pmatrix} \tag{7.4}$$

Here $x_t = (x \;\; y \;\; \theta)^T$ and $x_t = (x' \;\; y' \;\; \theta')^T$ are the state vectors at time $t-1$ and t, respectively. As before, the control is a vector of two independent velocities:

$$u_t \;\; = \;\; \begin{pmatrix} v_t \\ \omega_t \end{pmatrix}, \tag{7.5}$$

where v_t is a translational velocity, and ω_t is a rotational velocity, and Δt is the time window over which the robot motion is executed. The variable $\frac{v_t}{\omega_t}$ in (7.4) stands for the quotient $\frac{v_t}{\omega_t}$.

We already know from Chapter 3 that EKF localization maintains its local posterior estimate of the state, represented by the mean μ_{t-1} and covariance Σ_{t-1}. We also recall that the "trick" of the EKF lies in linearizing the motion and measurement model. For that, we decompose the motion model into a noise-free part and a random noise component with (approximately) zero mean.

$$\underbrace{\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}}_{x_t} \;\; = \;\; \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}}_{g(u_t, x_{t-1})} + \mathcal{N}(0, R_t) \tag{7.6}$$

This is of the form (3.50) defined in Chapter 3. We recall from that chapter that EKF linearization approximates g through a Taylor expansion:

$$g(u_t, x_{t-1}) \;\; \approx \;\; g(u_t, \mu_{t-1}) \; + \; G_t \, (x_{t-1} - \mu_{t-1}) \tag{7.7}$$

The function $g(u_t, \mu_{t-1})$ is simply obtained by replacing the exact state x_{t-1}—-which we do not know—by our expectation μ_{t-1}—which we know. The Jacobian G_t is the derivative of the function g with respect to μ_{t-1}, and evaluated at u_t and μ_{t-1}:

$$G_t \;\; = \;\; g'(u_t, \mu_{t-1}) \;\; = \;\; \begin{pmatrix} 1 & 0 & \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 1 & \frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 1 \end{pmatrix} \tag{7.8}$$

Here $\mu_{t-1,\theta}$ denotes the third component of the mean vector μ_{t-1}, which is the estimate of the rotation θ_t.

The reader may have noticed that we silently ignored the fact that the amount of noise in motion Σ_u depends on the magnitude of robot motion u_t. When implementing EKF localization, the covariance R_t is a function of the velocities v_t and ω_t and the mean μ_{t-1}. This issue was already discussed in more depth in Chapter 5, and for the sake of brevity we leave the design of an appropriate Σ_u to the reader as an exercise.

EKF localization also requires a linearized measurement model with additive Gaussian noise, as discussed back in Chapter 3. The measurement model for our feature-based maps shall be a variant of Equation (6.41) in Chapter 6.6 which presupposes knowledge of the landmark identity via the correspondence variables. x Let us denote by j the identity of the landmark that corresponds to the i-th component in the measurement vector. Then we have

$$\begin{aligned} z_t^i &= \begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{(m_{j,x}-x)^2 + (m_{j,y}-y)^2} \\ \text{atan2}(m_{j,y}-y, m_{j,x}-x) - \theta \\ m_{j,s} \end{pmatrix} + \begin{pmatrix} \mathcal{N}(0,\sigma_r) \\ \mathcal{N}(0,\sigma_\phi) \\ \mathcal{N}(0,\sigma_s) \end{pmatrix} \end{aligned} \tag{7.9}$$

Here $(m_{j,x} \;\; m_{j,y})^T$ are the coordinates of the landmark observed at time t, and $m_{j,s}$ is its (correct) signature. The variables r_t^i and ϕ_t^i denote the range and bearing to this landmark, and s_t^i denotes the signature as perceived by the robot. the variances of the measurement noise is given by σ_r, σ_ϕ, and σ_s, respectively. To bring this into a form familiar from Chapter‘3, we rewrite this model as follows:

$$z_t^i = h(x_t, j, m) + \mathcal{N}(0, Q_t) \tag{7.10}$$

with $x_t = (x \;\; y \;\; \theta)^T$ and

$$h(x_t, j, m) = \begin{pmatrix} \sqrt{(m_{j,x}-x)^2 + (m_{j,y}-y)^2} \\ \text{atan2}(m_{j,y}-y, m_{j,x}-x) - \theta \\ m_{j,s} \end{pmatrix} \tag{7.11}$$

and

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix} \tag{7.12}$$

The Taylor approximation follows now directly from Equation (3.52):

$$h(x_t, j, m) \quad \approx \quad h(\bar{\mu}_t, j, m) \; + \; H_t^i \; (x_t - \bar{\mu}_t) \tag{7.13}$$

Here H_t is the Jacobian of h at $\bar{\mu}_t$, calculated for the i-the landmark and the map m. This approximation substitutes the exact robot pose x_t in $h(x_t, j, m)$ by the estimate $\bar{\mu}_t = (\bar{\mu}_{t,x} \;\; \bar{\mu}_{t,y} \;\; \bar{\mu}_{t,\theta})^T$. The Jacobian of h is given by the following matrix

$$H_t^t \quad = \quad h'(\bar{\mu}_t, j, m) \quad = \quad \begin{pmatrix} \frac{\partial r_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial s_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial s_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial s_t^i}{\partial \bar{\mu}_{t,\theta}} \end{pmatrix} \tag{7.14}$$

in which $\bar{\mu}_t = (\bar{\mu}_{t,x} \;\; \bar{\mu}_{t,y} \;\; \bar{\mu}_{t,\theta})^T$ denotes the estimate $\bar{\mu}_t$ factored into its individual three values. Calculating the desired derivatives from Equation (7.11) gives us the following matrix:

$$H_t^i \quad = \quad \begin{pmatrix} \frac{m_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q}_t} & \frac{y_t - \bar{\mu}_{t,y}}{\sqrt{q}_t} & 0 \\ \frac{\bar{\mu}_{t,y} - y_t}{q_t} & \frac{m_{j,x} - \bar{\mu}_{t,x}}{q_t} & -1 \\ 0 & 0 & 0 \end{pmatrix} \tag{7.15}$$

with $q_t = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$, and $j = c_t^i$ if the landmark that corresponds to the measurement z_t^i. This linear approximation, plugged into the standard EKF equations, leads to lines 7 through 11 in the EKF localization algorithm shown in Table 7.2.

Finally, we note that our feature-based localizer processes multiple measurements at-a-time, whereas the EKF discussed in Chapter 3.2 only processed a single sensor item. Our algorithm relies on an implicit conditional independence assumption, which we briefly discussed in Chapter 6.6, Equation (6.40). Essentially, we assume that all feature measurement probabilities are independent given the pose x_t and the map m:

$$p(z_t \mid x_t, m) \quad = \quad \prod_i p(z_t^i \mid x_t, m) \tag{7.16}$$

This is usually a good assumption, especially if the world is static. It enables us to add the information from multiple features into our filter, as specified in lines 14 and 15 in Table 7.2. This slightly non-obvious addition in Table 7.2 stems from the fact that multiple features are integrated via Bayes rule (which is a product). The addition takes place in information space, in which probabilities are represented logarithmically (c.f., Chapter 3.4). The logarithm of a product is a sum, hence the additive form of this integration step.

7.6 ESTIMATING CORRESPONDENCES

7.6.1 EKF Localization with Unknown Correspondences

The EKF localization discussed thus is only applicable when landmark correspondences can be determined with absolute certainty. In practice, this is rarely the case. Most implementations therefore determine the identity of the landmark during localization. Throughout this book, we will encounter a number of strategies to cope with the correspondence problem. The most simple of all is known as *maximum likelihood* correspondence, in which one first determines the most likely value of the correspondence variable, and then takes this value for granted.

Maximum likelihood techniques are brittle if there are many equally likely hypotheses for the correspondence variable. However, one can often design the system for this to be not the case. To reduce the danger of asserting a false data association, there exist essentially two techniques: First, select landmarks that are sufficiently unique and sufficiently far apart from each other that confusing them with each other is unlikely. Second, make sure that the robot's pose uncertainty remains small. Unfortunately, these two strategies are somewhat counter to each other, and finding the right granularity of landmarks in the environment can be somewhat of an art.

Nevertheless, the maximum likelihood technique is of great practical importance. Table 7.3 depicts the EKF localization algorithm with a maximum likelihood estimator for the correspondence. The motion update in Lines 2 through 4 is identical to the one in Table 7.2. The key difference is in the measurement update: Here we first calculate for all landmarks k in the map a number of quantities that enables us to determine the most likely correspondence (Lines 6 through 12). The correspondence variable is then chosen in Line 14, by minimizing a quadratic Mahalanobis distance function defined over the measured feature vector z_t^i and the expected measurement $\hat{z}_t^k$, for any

1: **Algorithm EKF_localization(**$\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$**):**

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

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

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

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

6: *for all landmarks* k *in the map* m *do*

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

8: $q_k = \delta_k^T \delta_k$

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

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

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

12: *endfor*

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

14: $j(i) = \underset{k}{\text{argmin}}(z_t^i - \hat{z}_t^k)^T\ \Psi_k^{-1}\ (z_t^i - \hat{z}_t^k)$

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

16: *endfor*

17: $\mu_t = \bar{\mu}_t + \sum_i K_t^i\ (z_t^i - \hat{z}_t^{j(i)})$

18: $\Sigma_t = (I - \sum_i K_t^i\ H_t^{j(i)})\ \bar{\Sigma}_t$

19: *return* μ_t, Σ_t

Table 7.3 The extended Kalman filter (EKF) localization algorithm with unknown correspondences. The correspondences $j(i)$ are estimated via a maximum likelihood estimator.

possible landmark m_k in the map. The covariance in this expression is composed of the measurement uncertainty, calculated in Line 5, and the robot uncertainty projected into the measurement space (Line 11). The final EKF update in Lines 17 and 18 only incorporates the most likely correspondences.

The algorithm in Table 7.2 is inefficient. It can be improved through a more thoughtful selection of landmarks in Lines 6 through 12. In most settings, the robot only sees a small number of landmarks at a time in its immediate vicinity; and simple tests can reject a large number of unlikely landmarks in the map.

Further, the algorithm can be modified to accommodate outliers. The standard approach is to only accept landmarks for which the Mahalanobis distance in Line 14, or the associated probability, passes a threshold test. This is generally a good idea: Gaussians fall of exponentially, and a single outlier can have a huge effect on the pose estimate. In practice, thresholding adds an important layer of robustness to the algorithm without which EKF localization tends to be brittle.

7.6.2 Mathematical Derivation

The maximum likelihood estimator determines the correspondence that maximizes the data likelihood.

$$\hat{c}_t = \operatorname*{argmax}_{c_t} p(z_t \mid c_{1:t}, m, z_{1:t-1}, u_{1:t}) \tag{7.17}$$

Here c_t is the correspondence vector at time t. The vector $z_t = \{z_t^1, z_t^2, \ldots\}$ is the measurement vector which contains the list of features z_t^i observed at time t. As before, each feature vector now contains three elements, the range, the bearing, and the signature:

$$z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T \tag{7.18}$$

The argmax in (7.17) selects the correspondence vector $\hat{c}_t$ that maximized the likelihood of the measurement. Note that this expression is conditioned on prior correspondences $c_{1,t-1}$. While those have been estimated in previous update steps, the maximum likelihood approach treats those as if they are always correct. This has two important ramifications: It makes it possible to update the filter incrementally. But it also introduces brittleness in the filter, which tends to diverge when correspondence estimates is erroneous.

The likelihood $p(z_t \mid c_{1:t}, m, z_{1:t-1}, u_{1:t})$ in Equation (7.17) is now easily computed from the belief $\overline{bel}(x_t)$ by integrating over the pose x_t, and omitting irrelevant conditioning variables:

$$\begin{aligned} & p(z_t \mid c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ &= \int p(z_t \mid c_{1:t}, x_t, m, z_{1:t-1}, u_{1:t})\; p(x_t \mid c_{1:t}, m, z_{1:t-1}, u_{1:t})\; dx_t \\ &= \int p(z_t \mid c_t, x_t, m)\; p(x_t \mid c_{1,t-1}, m, z_{1:t-1}, u_{1:t})\; dx_t \\ &= \int p(z_t \mid c_t, x_t, m)\; \overline{bel}(x_t)\; dx_t \end{aligned} \tag{7.19}$$

This yields the following maximization

$$\hat{c}_t = \operatorname*{argmax}_{c_t} \int p(z_t \mid c_t, x_t, m)\; \overline{bel}(x_t)\; dx_t \tag{7.20}$$

This maximization is carried out over the entire correspondence vector c_t.

Unfortunately, there are exponentially many terms in this maximization. When the number of features per measurement is large, the number of possible feature vectors may grow too large for practical implementations. The most common technique to avoid such an exponential complexity performs the maximization separately for each individual feature z_t^i in the measurement vector $z_t = \{z_t^1, z_t^2, \ldots\}$.

$$\hat{c}_t^i = \operatorname*{argmax}_{c_t^i} \int p(z_t^i \mid c_t^i, x_t, m)\; \overline{bel}(x_t)\; dx_t \tag{7.21}$$

The implications of this component-wise optimization will be discussed below; we note that it is "justified" only when we happen to know that individual feature vectors are conditionally independent—an assumption that is usually adopted for convenience, and that we will discuss further below.

$$p(z_t \mid c_t, x_t, m) = \prod_i p(z_t^i \mid c_t, x_t, m) \tag{7.22}$$

This assumption is analogous to the one stated in Equation (6.40). Under this assumption, the term that is being maximized in (7.20) becomes a product of terms with

disjoint optimization parameters, for which the maximum is attained when each individual factor is maximal.

The measurement probability for each z_t^i is given by the following exponential function:

$$\begin{aligned} & p(z_t^i \mid c_t^i, x_t, m) \\ & = \eta \, \exp\left\{-\tfrac{1}{2}(z_t^i - h(x_t, c_t^i, m))^T \, Q_t^{-1} \, (z_t^i - h(x_t, c_t^i, m))\right\} \end{aligned} \tag{7.23}$$

where h is defined as in (7.11). Applying once again our Taylor expansion as in (3.52) gives us

$$\begin{aligned} p(z_t^i \mid c_t^i, x_t, m) \quad \approx \quad & \eta \, \exp\left\{-\tfrac{1}{2}(z_t^i - h(\bar{\mu}_t, c_t^i, m) \; - \; H_t \, (x_t - \bar{\mu}_t))^T \, Q_t^{-1}\right. \\ & \left. (z_t^i - h(\bar{\mu}_t, c_t^i, m) \; - \; H_t \, (x_t - \bar{\mu}_t))\right\} \end{aligned} \tag{7.24}$$

Thus, the integral (7.21) can be written as

$$\int \eta \, \exp\{-L_t(z_t^i, c_t^i, x_t, m)\} \, dx_t \tag{7.25}$$

with

$$\begin{aligned} L_t(z_t^i, c_t^i, x_t, m) \quad = \quad & \tfrac{1}{2}(z_t^i - h(\bar{\mu}_t, c_t^i, m) \; - \; H_t \, (x_t - \bar{\mu}_t))^T \, Q_t^{-1} \\ & (z_t^i - h(\bar{\mu}_t, c_t^i, m) \; - \; H_t \, (x_t - \bar{\mu}_t)) \\ & + \; \tfrac{1}{2} \, (x_t - \bar{\mu}_t)^T \, \bar{\Sigma}_t^{-1} \, (x_t - \bar{\mu}_t) \end{aligned} \tag{7.26}$$

We already encountered integrals of this form in Chapter 3.2, where we derived the motion update of the Kalman filter and the EKF. The closed-form solution to this integral is derived completely analogously to those derivations. In particular, the Gaussian defined by (7.25) has mean $h(\bar{\mu}_t, c_t^i, m)$ and covariance $H_t \, \bar{\Sigma}_t \, H_t^T + Q_t$. Thus, we have under our linear approximation the following closed form expression for the measurement likelihood:

$$\int p(z_t^i \mid c_t^i, x_t, m) \, \overline{bel}(x_t) \, dx_t \quad \sim \quad \mathcal{N}(h(\bar{\mu}_t, c_t^i, m), H_t \, \bar{\Sigma}_t \, H_t^T + Q_t) \tag{7.27}$$

and thus

$$\begin{aligned} & p(z_t^i \mid c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ &= \eta \, \exp\left\{-\tfrac{1}{2} \, (z_t^i - h(\bar{\mu}_t, c_t^i, m))^T \, [H_t \, \bar{\Sigma}_t \, H_t^T + Q_t]^{-1} \, (z_t^i - h(\bar{\mu}_t, c_t^i, m))\right\} \end{aligned} \tag{7.28}$$

Since we only seek the value for c_t^i that maximizes this function, it suffices to maximize the quadratic term in the exponential:

$$\hat{c}_t^i = \operatorname*{argmax}_{c_t^i} \, (z_t^i - h(\bar{\mu}_t, c_t^i, m))^T \, [H_t \, \bar{\Sigma}_t \, H_t^T + Q_t]^{-1} \, (z_t^i - h(\bar{\mu}_t, c_t^i, m)) \tag{7.29}$$

This calculation is implemented in Line 15 in Table 7.3.

This distribution is remarkably similar to the probability calculated by the algorithm **landmark_model_known_correspondence** in Table 6.4, the only difference arising from the covariance term. In Table 6.4, the robot pose is assumed to be known, hence the covariance reduces to the measurement covariance Q_t. Here we only have a probabilistic estimate of the pose, hence have to use our best estimate for the pose, and fold in the uncertainty in our estimate. Our state estimate is given by $\bar{\mu}_t$. As the derivation above shows, the covariance is adjusted by the additional uncertainty $H_t \, \bar{\Sigma}_t \, H_t^T$, which is the projection of the pose uncertainty under the linearized approximation of the measurement function h. This shows the correctness of the calculation in lines 12 and 13 in our EKF algorithm in Table 7.3: π_k is indeed the desired likelihood. The correctness of the algorithm follows from our assumption that feature measurements are conditionally independent, as stated in Equation (7.22). Of course, this independence is usually violated, which makes our algorithm approximate. A further approximation is introduced by the Taylor expansion.

7.7 MULTI-HYPOTHESIS TRACKING

There exist a number of extensions of the basic EKF to accommodate situations where the correct data association cannot be determined with sufficient reliability. Several of those techniques will be discussed later in this book, hence our exposition at this point will be brief.

A classical technique that overcomes difficulties in data association is the *Multi-hypothesis Tracking Algorithm (MHT)*. The MHT can represent a belief by multiple

Gaussians, that is, the posterior is represented by the mixture

$$bel(x_t) \quad = \quad \frac{1}{\sum_l \psi_{t,l}} \sum_l \psi_{t,l} \det\left(2\pi\Sigma_{t,l}\right)^{-\frac{1}{2}} \; \exp\left\{-\tfrac{1}{2}(x_t - \mu_{t,l})^T \Sigma_{t,l}^{-1}(x_t - \mu_{t,l})\right\} \tag{7.30}$$

Here l is the index of the mixture component. Each such component, or "track" in MHT slang, is itself a Gaussian with mean $\mu_{t,l}$ and covariance $\Sigma_{t,l}$. The scalar $\psi_{t,l} \geq 0$ is a *mixture weight*. It determines the weight of the l-th mixture component in the posterior. Since the posterior is normalized by $\sum_l \psi_{t,l}$, each $\psi_{t,l}$ is a relative weight, and the contribution of the l-th mixture component depends on the magnitude of all other mixture weights.

As we shall see below when we describe the MHT algorithm, each mixture component relies on a unique sequence of data association decisions. Hence, it makes sense to write $c_{t,l}$ for the data association vector associated with the l-th track, and $c_{1:t,l}$ for all past and present data associations associated with the l-th mixture component. With this notation, we can now think of mixture components as contributing local belief functions conditioned on a unique sequence of data associations:

$$bel_l(x_t) \quad = \quad p(x_t \mid z_{1:t}, u_{1:t}, c_{1:t,l}) \tag{7.31}$$

Here $c_{1:t,l} = \{c_{1,l}, c_{2,l}, \ldots, c_{t,l}\}$ denotes the sequence of correspondence vectors associated with the l-th track.

Before describing the MHT, it makes sense to discuss a completely intractable algorithm from which the MHT is derived. This algorithm is the full Bayesian implementation of the EKF under unknown data association. It is amazingly simple: Instead of selecting the most likely data association vector, our fictitious algorithm maintains them all. More specifically, at time t each mixture is split into many new mixtures, each conditioned on a unique correspondence vector c_t. Let m be the index of one of the new Gaussians, and l be the index from which this new Gaussian is derived, for the correspondence $c_{t,l}$. The weight of this new mixture is then set to

$$\psi_{t,m} \quad = \quad \psi_{t,l} \; p(z_t \mid c_{1:t-1,l}, c_{t,m}, z_{1:t-1}, u_{1:t}) \tag{7.32}$$

This is the product of the mixture weight $\psi_{t,l}$ from which the new component was derived, times the likelihood of the measurement z_t under the specific correspondence

vector that led to the new mixture component. In other words, we treat correspondences as latent variable and calculate the posterior likelihood that a mixture component is correct. A nice aspect of this approach is that we already know how to compute the measurement likelihood $p(z_t \mid c_{1:t-1,l}, c_{t,m}, z_{1:t-1}, u_{1:t})$ in Equation (7.32): It is simply the product $\prod \pi_k$ of the individual feature likelihoods computed in line 13 of our localization algorithm in Table 7.3. Thus, we can incrementally calculate the mixture weights for each new component. The only downside of this algorithm is the fact that the number of mixture components, or tracks, grow exponentially over time.

The MHT algorithm approximates this algorithm by keeping the number of mixture components small. In essence, it terminates every component whose relative mixture weight

$$\frac{\psi_{t,l}}{\sum_m \psi_{t,m}} \tag{7.33}$$

is smaller than a threshold $\psi_{\min}$. It is easy to see that the number of mixture components is always at most $\psi_{\min}^{-1}$. Thus, the MHT maintains a compact posterior that can be updated efficiently. It is approximate in that it maintains a very small number of Gaussians, but in practice the number of plausible robot locations is usually very small.

We will omit a formal description of the MHT algorithm at this point, and instead refer the reader to a large number of related algorithms in this book. However, when implementing the MHT, it is useful to devise strategies for identifying low-likelihood tracks before instantiating them.

7.8 PRACTICAL CONSIDERATIONS

The EKF localization algorithm and its close relative, MHT localization, are popular techniques for position tracking. There exist a large number of variations of these algorithm that enhance their efficiency and robustness.

- **Efficient search.** First, it is often impractical to loop through all landmarks k in the map. Often, there exist simple tests to identify plausible candidate landmarks (e.g., by simply projecting the measurement into x-y-space), enabling one to rule out all but a constant number of candidates. Such algorithms can be orders of magnitude faster that our naive implementations.

- **Mutual exclusion.** A key limitation of our implementations arises from our assumed independence of feature noise in the EKF (and, by inheritance, the MHT). The reader may recall condition (7.22), which enabled us to process individual features sequentially, thereby avoiding a potential exponential search through the space of all correspondence vectors. Unfortunately, such an approach allows for assigning multiple observed features, say z_t^i and z_t^j with $i \neq j$, to be assigned to the same landmark in the map: $\hat{c}_t^i = \hat{c}_t^i$. For many sensors, such a correspondence assignment is wrong by default. For example, if the feature vector is extracted from a single camera image, we know by default that two different regions in the image space must correspond to different locations in the physical world. Put differently, we usually know that $i \neq j \longrightarrow \hat{c}_t^i \neq \hat{c}_t^i$. This (hard!) constraint is called *mutual exclusion principle in data association.* It reduces the space of all possible correspondence vectors. Advanced implementations consider this constraint. For example, one might first search for each correspondence separately—as in our version of the EKF localizer—followed by a "repair" phase in which violations of the mutual exclusion principle are resolved by changing correspondence values accordingly.

- **Outliers.** Further, our implementation does not address the issue of outliers. The reader may recall from Chapter 6.6 that we allow for a correspondence $c = N + 1$, with N being the number of landmarks in the map. Such an outlier test is quite easily added to our algorithms. In particular, if we set π_{N+1} to be the a prior probability of an outlier, the argmax-step in line 15 EKF localization (Table 7.3) will default to $N + 1$ if an outlier is the most likely explanation of the measurement vector. Clearly, an outlier does not provide any information on the robot's pose; hence, the corresponding terms are simply omitted in lines 18 and 19 in Table 7.3.

Both the EKF and the MHT localization are only applicable to position tracking problems. In particular, linearized Gaussian techniques tend to work well only if the position uncertainty is small. There are three complimentary reasons for this observation:

- A uni-modal Gaussian is usually a good representation of uncertainty in tracking whereas it is not in more general global localization problems. Both the EKF and the MHT start with a single unimodal Gaussian, although the MHT can potentially branch into multiple local Gaussian.

- A narrow Gaussian reduces the danger of erroneous correspondence decisions. This is important particularly for the EKF, since a single false correspondence can derail the tracker by inducing an entire stream localization and correspondence errors. The MHT is more robust to this problem, though it can fail equally

when the correct correspondence is not among those maintained in the Gaussian mixture.

- The Taylor expansion is usually only good in a close proximity to the linearization point. As a rule of thumb, if the standard deviation for the orientation θ is larger then ± 20 degrees, linearization effects are likely to make the algorithm fail. This problem applies equally to the EKF and the MHT and explains why starting the MHT with a very wide initial Gaussian does not turn it into a global localization algorithm.

For all those reasons, the Gaussian localization algorithms discussed in this chapter are inapplicable to global localization problems or the kidnapped robot problem.

The design of the appropriate features for EKF localization is a bit of an art. This is because multiple competing objectives have to be met. On the one hand, one wants sufficiently many features in the environment, so that the uncertainty in the robot's pose estimate can be kept small. Small uncertainty is absolutely vital for reasons already discussed. On the other hand, one wants to minimize chances that landmarks are confused with each other, or that the landmark detector detects spurious features. Many environments do not possess too many point landmarks that can be detected with high reliability, hence many implementation rely on relatively sparsely distributed landmarks. Here the MHT has a clear advantage, in that it is more robust to data association errors. As a rule of thumb, large numbers of landmarks tend to work better than small numbers even for the EKF. When landmarks are dense, however, it is critical to apply the mutual exclusion principle in data association.

Finally, we note that EKF localization processes only a subset of all information in the sensor measurement. By going from raw measurements to features, the amount of information that is being processed is already drastically reduced. Further, EKF localization is unable to process *negative* information. Negative information pertains to the absence of a feature. Clearly, not seeing a feature when one expects to see it carries relevant information. For example, not seeing the Eiffel Tour in Paris implies that it is unlikely that we are right next to it. The problem with negative information is that it induces non-Gaussian beliefs, which cannot be represented by the mean and variance. For this reason, EKF implementations simply ignore the issue of negative information, and instead integrate only information from observed features. The standard MHT also avoids negative information. However, it is possible to fold negative information into the mixture weight, by decaying mixture components that failed to observe a landmark.

With all these limitations, does this mean that Gaussian techniques are generally brittle and inapplicable to more general localization techniques? The answer is no. In fact,

the key to successful localization lies in the approach for data association. Later in this book, we will encounter more sophisticated techniques for handling correspondences than the ones discussed thus far. Many of these techniques are applicable (and will be applied!) to Gaussian representations, and the resulting algorithms are often among the best ones known.

7.9 SUMMARY

In this chapter, we introduced the mobile robot localization problem and devised a first practical algorithm for solving it.

- The localization problem is the problem of estimating a robot's pose relative to a known map of its environment.
- Position tracking addresses the problem of accommodating the local uncertainty of a robot whose initial pose is known; global localization is the more general problem of localizing a robot from scratch. Kidnapping is a localization problem in which a well-localized robot is secretly teleported somewhere else without being told—it is the hardest of the three localization problems.
- The hardness of the localization problem is also a function of the degree to which the environment changes over time. All algorithms discussed thus far assume a static environment.
- Passive localization approaches are filters: they process data acquired by the robot but do not control the robot. Active techniques control the robot. In this and the next chapter, we study passive approaches; active approaches will be discussed in Chapter **??** of this book.
- Markov localization is just a different name for the Bayes filter applied to the mobile robot localization problem.
- EKF localization applies the extended Kalman filter to the localization problem. EKF localization is primarily applied to feature-based maps.
- The most common technique for dealing with correspondence problems is the maximum likelihood technique. This approach simply assumes that at each point in time, the most likely correspondence is correct.
- The multi hypothesis tracking algorithm (MHT) pursues multiple correspondences, using a Gaussian mixture to represent the posterior. mixture components

are created dynamically, and terminated if their total likelihood sinks below a user-specified threshold.

- The MHT is more robust to data association problems than the EKF, at an increased computational cost.
- Both EKF and MHT localization are well-suited for local position tracking problems with limited uncertainty and in environments with distinct features. They are less applicable to global localization or in environments where most objects look alike.
- Selecting features for EKFs and MHTs requires skill! The performance of both algorithms can be improved by a number of measures, such as enforcing mutual exclusion in data association.

In the next chapter, we will discuss probabilistic localization techniques that can cope with more general localization problems, such as the global localization problem or the problem of localizing in dynamic environments.