9 *Bayesian Methods*

OPERATING IN the real world, robots lack the perfect sensors and deterministic actions of many artificial worlds. Rather, robots are faced with various kinds of uncertainty. In this chapter we continue to discuss probabilistic frameworks for typical fundamental tasks of mobile robots such as localization, mapping, and simultaneous localization and mapping (SLAM). While the methods presented in this chapter employ the same iterative prediction-update process that is used in the Kalman filter (see chapter 8), they do not rely on the restrictive assumptions required by the Kalman filter. The methods described here can use nonlinear models for both robot motion and sensing. Most important, the resulting estimate may be an arbitrary distribution instead of a Gaussian. Throughout this chapter we present the key ideas of successful techniques together with a derivation of their mathematical foundations. We will also discuss ways to efficiently implement these approaches, since the capability to represent arbitrary distributions can lead to higher computational demands compared to Kalman filters.

9.1 Localization

In the previous chapter, we achieved localization by maintaining a distribution of the robot by iteratively estimating the mean and covariance matrix of a Gaussian distribution. This way of representing a belief about the location of the robot assumes that there is no "ambiguity" in the sense that the distribution is always unimodal or more specifically a Gaussian. One form of localization in which this assumption is often met is *position tracking,* which assumes that the initial configuration of the robot

is (approximately) known and whose task is to keep track of the robot's location while it is moving through the environment. If the robot's configuration is approximately known and if there is only a small region of uncertainty around the true location of the robot, the observations of the robot can usually be associated uniquely with the corresponding features in its map. Consider, e.g., that a robot knows its location up to a few centimeters. If it detects a door, it can use this observation to accurately compute its location given the door stored in its map of the environment. If, however, the uncertainty is high and the robot knows its location only up to several meters, there might be multiple doors in the map that its current observation can correspond to. Accordingly, the situation is ambiguous and a single Gaussian obviously cannot appropriately represent the robot's belief about its location.

In this chapter, we consider a form of position estimation where the robot may have ambiguity, i.e., the belief about its location can be modeled by a multimodal distribution. The techniques described in this chapter are able to deal with a more complex version of localization called *global localization*. Here the robot has to estimate its location under global uncertainty as it is not given its initial location. The techniques can also solve the most complex problem of robot localization, the so-called *kidnapped robot problem*. The kidnapped robot problem, or the relocalization problem, is more complicated than the global localization problem because the robot has generated a false belief of its most likely location which it must identify and "unlearn" before it can relocalize.

9.1.1 The Basic Idea of Probabilistic Localization

Before we delve into mathematical detail, let us illustrate the basic concepts with a simple example. Consider the environment depicted in figure 9.1. For the sake of simplicity, assume that the space of robot locations is one-dimensional, i.e., the robot can only move horizontally. Now suppose the robot is switched on somewhere in this environment to start its operation, but it is not told its location. Probabilistic localization represents this state of uncertainty by a *uniform distribution* over all locations, as shown by the graph in the top diagram in figure 9.1. Now assume the robot queries its sensors and finds out that it is next to a door. Probabilistic localization modifies the belief by raising the probability for locations next to doors, and lowering it elsewhere. This is illustrated in the second diagram in figure 9.1. Notice that the resulting belief is multimodal, reflecting the fact that the available information is insufficient to uniquely derive the robot's configuration. Also note that locations not close to a door still possess nonzero probability. This is because sensor readings are noisy, and a single sight of a door is typically insufficient to exclude the possibility of not being next to a door.

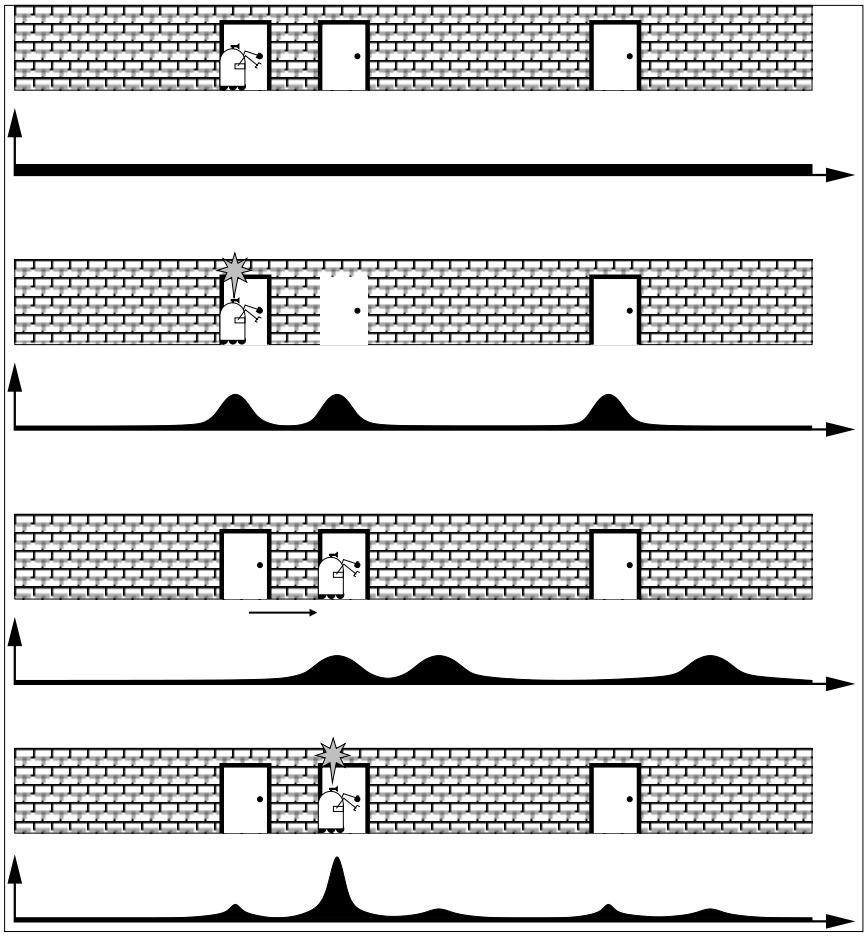

Figure 9.1 The basic idea of probabilistic localization: a mobile robot during global localization.

Now the robot advances to the next door. Probabilistic localization incorporates this information by propagating the belief distribution accordingly. To account for the inherent noise in robot motion, which in this situation inevitably leads to a loss of information, the new belief is smoother (and less certain) than the previous one. This is visualized in the third diagram in figure 9.1. Finally, the robot senses a second time, and again finds itself next to a door. This observation is combined with the current (nonuniform) belief, which leads to the final belief shown in the bottom diagram in figure 9.1. At this point, "most" of the probability is centered around a single location. The robot is now quite certain about its location.

Note that the final belief includes five different peaks given our sequence of two observations and one motion. The four smaller peaks correspond to the four cases in which the robot could only once explain its two observations given the map of the environment. At the location of the highest peak, which is in front of the second door at the true location of the robot, the robot has correctly identified a door twice. All other locations have small probabilities, since the robot could not explain its observations using its map.

Note that in this example the robot did not have an erroneous measurement. A false-positive detection of a door would lead to a situation in which the highest peak does not correspond to the true location of the robot. If, however, the robot knows about potential measurement errors, it would not become overly confident by just a few observations. One of the key features of probabilistic localization is that it uses the sensory information obtained to compute a belief that most accurately reflects the uncertainty about the configuration of the robot, given the knowledge about the behavior of the sensors of the robot.

Moreover, if the doors were uniquely identifiable by the robot, a Kalman filter would be sufficient for global localization. Since the robot is not able to identify the door it has sensed, it cannot associate an observation of a door uniquely to the doors given in its map. This problem is well-known as the *data association* problem. If the data association is known, Kalman filters can in fact be sufficient. Without knowing how to associate measurements to features in the map, the resulting beliefs will be inherently multimodal due to the resulting ambiguities. The strength of probabilistic localization lies in its capability to allow the representation of arbitrary distributions that are much more flexible than Gaussians. Put another way, probabilistic localization can also be applied when the data association is unknown or when the robot's motion models or sensor models are highly nonlinear.

9.1.2 Probabilistic Localization as Recursive Bayesian Filtering

Let X be the state space for the robot. We want to estimate the state $x \in X$ of the robot, which essentially is its configuration given as its position and orientation. In probabilistic localization the robot estimates at every time step k the conditional probability $P(x(k) \mid u(0 : k-1), y(1 : k))$ over all possible configurations given the sensor information $y(1 : k)$ it gathered about the environment and the movements $u(0 : k-1)$ carried out. The term $y(1 : k)$ denotes all observations obtained in the time steps $1, \ldots, k$. The notation $y(1 : k)$ "unfolds" to $y(1), y(2), \ldots, y(k)$ and $u(0 : k-1)$ unfolds in a similar fashion. The term $P(x(k) \mid u(1 : k-1), y(1 : k))$ is usually called the *posterior probability* (or simply *posterior*) [347]. Note that when u and y are written side by side, we assume that data have arrived in a synchronized way,

i.e., in the form $u(0), y(1), \ldots, u(k-1), y(k)$. This assumption makes the derivation of probabilistic localization easier, but our algorithms can easily be extended to data streams that are not synchronized.

Note that in the previous chapter, $u(k)$ was simply the control input. In this chapter, we denote it as movements and do not rely on a specific interpretation of $u(k)$. It can represent the commanded velocities, the odometry measurements, or the result of filtering and fusing commanded velocities and odometry measurements, as described in the previous chapter.

Throughout this section we will assume that the robot is also given a model or map m of the environment. In principle we have to add this model as background knowledge in every term. However, for the sake of simplicity we will skip m in the equations below and assume that it is given as background knowledge. The heart of probabilistic localization is the following equation which tells us how to use the sensory input to update the most recent estimate (the *prior*) to obtain a new estimate (the posterior):

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k)) \\ &\quad = \eta(k)\ P(y(k) \mid x(k)) \sum_{x(k-1) \in X} (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad P(x(k-1) \mid u(0:k-2), y(1:k-1))) \end{aligned} \tag{9.1}$$

As mentioned above, the term $P(x(k) \mid u(0:k-1), y(1:k))$ is the posterior about the location of the robot at time k given the input data gathered so far. The term $P(x(k-1) \mid u(0:k-2), y(1:k-1)))$, in contrast, is denoted as the *prior* as it quantifies the probability that the robot is at location $x(k-1)$ before the integration of $u(k-1)$ and $y(k)$. The term $P(y(k) \mid x(k))$ is called the *observation model* which specifies the likelihood of the measurement $y(k)$ given the robot is at location $x(k)$. The term $P(x(k) \mid u(k-1), x(k-1))$ represents the *motion model* and can be regarded as a transition probability. It specifies the likelihood that the movement action $u(k-1)$ carried out at location $x(k-1)$ carries the robot to the location $x(k)$. Finally, $\eta(k)$ is a *normalization constant* that ensures that the left-hand side of this equation sums up to one over all $x(k)$. Note that (9.1) effectively accomplishes a combination of both the prediction and update steps of the Kalman filter.

Equation (9.1) is a special case of the following general equation for recursive Bayesian filtering.

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k)) \\ &\quad = \eta(k)\ P(y(k) \mid x(k)) \int_X (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad P(x(k-1) \mid u(0:k-2), y(1:k-1)))\, dx(k-1) \end{aligned} \tag{9.2}$$

Whereas (9.1) assumes discrete state spaces, (9.2) deals with continuous state spaces. Additionally, (9.2) can be shown to be a generalization of Kalman filtering. In this context the term $P(x(k) \mid u(k-1), x(k-1))$ is a generalization of (8.1) (see chapter 8) to arbitrary and nonlinear noise. Similarly, the term $P(y(k) \mid x(k))$ can handle arbitrary and nonlinear noise in the measurements. Finally, the posterior $P(x(k) \mid u(0:k-1), y(1:k))$ generalizes the belief representation of Kalman filters from Gaussians to arbitrary probability density functions (PDFs).

Note the recursive character of probabilistic localization. The belief at time k is computed out of the posterior at time $k-1$ by incorporating two quantities, namely $P(y(k) \mid x(k))$ and $P(x(k) \mid u(k-1), x(k-1))$. Obviously, both the motion model and the observation model are the crucial components of probabilistic localization. Further below we will describe typical realizations of these models. We will also discuss different ways to represent the posterior $P(x(k) \mid u(k-1), x(k-1))$ and describe how to update the posterior given these representations.

Independent of the specific representation, the update of the belief is generally carried out in two different steps. The two steps are the *prediction step* and the *update step* which are joined together by (9.1). Note that the separation of a filtering process into these two steps is common in the context of Kalman filtering. The prediction step is applied whenever the belief has to be updated because of an odometry measurement $u(k-1)$. Suppose $u(0:k-2)$ and $y(1:k-1)$ are the data obtained thus far and $P(x(k-1) \mid u(0:k-2), y(1:k-1)))$ is the current belief about the configuration of the robot. Then we obtain the resulting belief $P(x(k) \mid u(0:k-1), y(1:k-1))$ by integrating over all possible previous configurations $x(k-1)$. For each such $x(k-1)$ we multiply $P(x(k-1) \mid u(0:k-2), y(1:k-1))$ by the probability $P(x(k) \mid u(k-1), x(k-1))$ that the measured motion action $u(k-1)$ has carried the robot from $x(k-1)$ to $x(k)$ and compute $P(x(k) \mid u(0:k-1), y(1:k-1))$ as the sum over all these values, i.e.,

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k-1)) \\ &\quad = \sum_{x(k-1) \in X} (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad\qquad P(x(k-1) \mid u(0:k-2), y(1:k-1))) \end{aligned} \tag{9.3}$$

Note that this operation basically corresponds to the step depicted in the third diagram of figure 9.1.

The update step is carried out whenever the robot perceives a measurement $y(k)$ with information about its environment. Suppose the current belief of the robot is $P(x(k) \mid u(0:k-1), y(1:k-1))$. In the update step we simply multiply for each configuration $x(k)$ the current value $P(x(k) \mid u(0:k-1), y(1:k-1))$ with the likelihood

of $P(y(k) \mid x(k))$ that the robot perceives $y(k)$ given the map of the environment and given that the robot's configuration is $x(k)$. Additionally, we multiply each value with a normalization constant that ensures that $P(x(k) \mid u(0:k-1),\, y(1:k))$ sums up to one over all $x(k)$, i.e.,

$$\begin{aligned} &P(x(k) \mid u(0:k-1),\, y(1:k)) \\ &\quad = \eta(k)\; P(y(k) \mid x(k))\; P(x(k) \mid u(0:k-1),\, y(1:k-1)). \end{aligned} \tag{9.4}$$

According to Bayes rule, the constant $\eta(k)$ is given as

$$\eta(k) = P(y(k) \mid u(0:k-1),\, y(1:k-1))^{-1}, \tag{9.5}$$

which generally is hard to compute. This is mainly because the dependency between consecutive measurements without any information about the location of the robot in general is hard to determine. However, if we apply the law of total probability, we can sum over all locations $x(k)$ and transform (9.5) to

$$\eta(k) = \left[\sum_{x(k) \in X} P(y(k) \mid x(k))\; P(x(k) \mid u(0:k-1),\, y(1:k-1)) \right]^{-1}. \tag{9.6}$$

Obviously, we now can compute $\eta(k)$ using the terms that are already contained in (9.4). As we will see later, using this equation the normalization constant can be computed on the fly while integrating $y(k)$ into the current belief.

To summarize, we have the following equations that completely describe the two individual steps of recursive Bayesian filtering and that correspond to the prediction and update steps also found in the Kalman filter:

prediction:

$$\begin{aligned} &P(x(k) \mid u(0:k-1),\, y(1:k-1)) \\ &\quad = \sum_{x(k-1) \in X} \big(P(x(k) \mid u(k-1),\, x(k-1)) \\ &\qquad\qquad P(x(k-1) \mid u(0:k-2),\, y(1:k-1))\big) \end{aligned}$$

update:

$$\begin{aligned} &\eta(k) \\ &\quad = \left[\sum_{x(k) \in X} P(y(k) \mid x(k))\; P(x(k) \mid u(0:k-1),\, y(1:k-1)) \right]^{-1} \\ &P(x(k) \mid u(0:k-1),\, y(1:k)) \\ &\quad = \eta(k)\; P(y(k) \mid x(k))\; P(x(k) \mid u(0:k-1),\, y(1:k-1)). \end{aligned}$$

In probabilistic localization the initial belief $P(x(0))$, reflects the prior knowledge about the initial configuration of the robot. This distribution can be initialized arbitrarily, but in practice two cases prevail. If the configuration of the robot relative to its map is entirely unknown, $P(x(0))$ is usually uniformly distributed, or if the initial state of the robot would be known up to a slight uncertainty, one would initialize $P(x(0))$ using a narrow Gaussian distribution centered at the robot's believed configuration.

The reader may notice that the principle of probabilistic localization leaves open

1. how the belief $P(x)$ is represented as well as
2. how the conditional probabilities $P(x(k) \mid u(k-1), x(k-1))$ and $P(y(k) \mid x(k))$ are computed.

Accordingly, existing approaches to probabilistic localization mainly differ in the representation of the belief and the way the perceptual and motion models are represented. After a derivation of the equation for probabilistic localization in the following subsection, we will discuss different ways to represent the posterior. As we will see, the representation of the posterior has a serious impact on the efficiency of probabilistic localization and the type of situations that can be accommodated with probabilistic localization.

9.1.3 Derivation of Probabilistic Localization

When computing $P(x \mid u(0:k-1),\, y(1:k))$, we distinguish two cases, depending on whether the most recent data item is an odometry reading or a sensor measurement.[1] Let us first consider how to incorporate the most recent data item, namely a sensor measurement $y(k)$ the robot uses to gather information about its environment. If we apply Bayes rule considering $y(1:k-1)$ and $u(0:k-1)$ as background knowledge, we obtain

$$\begin{aligned} &P(x(k) \mid u(0:k-1),\, y(1:k)) \\ &= \frac{P(y(k) \mid u(0:k-1),\, y(1:k-1),\, x(k))\, P(x(k) \mid u(0:k-1),\, y(1:k-1))}{P(y(k) \mid u(0:k-1),\, y(1:k-1))}. \end{aligned} \tag{9.7}$$

First consider the left term $P(y(k) \mid u(k-1),\, y(1:k-1),\, x(k))$ in the numerator. This term represents the likelihood of the most recent measurement $y(k)$ given all previous measurements and given that the configuration $x(k)$ of the robot at time k is known. In recursive Bayesian filtering, one generally makes the assumption that, once the state $x(k)$ is known, the measurement $y(k)$ is independent of all previous measurements and controls. Given this assumption we can simply remove $y(1:k-1)$

1. These two cases are analogous to the Kalman prediction and update steps, respectively.

and $u(0:k-1)$ from this term. Accordingly, we simplify (9.7) to

$$P(x(k) \mid u(0:k-1), y(1:k)) = \frac{P(y(k) \mid x(k))\ P(x(k) \mid u(0:k-1), y(1:k-1))}{P(y(k) \mid u(0:k-1), y(1:k-1))}. \tag{9.8}$$

Observe that the denominator is a normalizer that does not depend on the configuration of the robot. It simply ensures that the left-hand side of (9.8) sums up to one over all $x(k)$. Accordingly, we can replace the denominator by a normalization constant $\eta(k)$ which is the same for all $x(k)$. This leads to

$$P(x(k) \mid u(0:k-1), y(1:k)) = \eta(k)\ P(y(k) \mid x(k))\ P(x(k) \mid u(0:k-1), y(1:k-1)). \tag{9.9}$$

To see how to incorporate the motions of the robot into the belief we next consider the rightmost term $P(x(k) \mid u(0:k-1), y(1:k-1))$ in this equation. If we use the law of total probability we derive

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k-1)) \\ &\quad = \sum_{x(k-1) \in X} \big[P(x(k) \mid u(0:k-1), y(1:k-1), x(k-1)) \\ &\qquad\qquad P(x(k-1) \mid u(0:k-1), y(1:k-1))\big]. \end{aligned} \tag{9.10}$$

To simplify the lefthand term in the sum we again make an independence assumption. We assume that $x(k)$ is independent of the measurements $y(1:k-1)$ and the movements $u(0:k-2)$ obtained and carried out before the robot arrived at $x(k-1)$ given we know $x(k-1)$. Rather the likelihood of being at $x(k)$ only depends on $x(k-1)$ and the most recent movement $u(k-1)$, i.e.,

$$P(x(k) \mid u(0:k-1), y(1:k-1), x(k-1)) = P(x(k) \mid u(k-1), x(k-1)) \tag{9.11}$$

Thus we simplify (9.10) to

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k-1)) \\ &\quad = \sum_{x(k-1) \in X} (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad\qquad \cdot P(x(k-1) \mid u(0:k-1), y(1:k-1))). \end{aligned} \tag{9.12}$$

Now consider the second factor $P(x(k-1) \mid u(0:k-1), y(1:k-1))$ in the sum. This term specifies the probability that the robot's configuration at time $k-1$ is $x(k-1)$ given the motions $u(0:k-1)$ and given the observations $y(1:k-1)$.

According to our terminology, the motion $u(k-1)$ is carried out at time step $k-1$ so that $u(k-1)$ carries the robot away from $x(k-1)$. Since we have no information about $x(k)$ and under the assumption that the time that elapses between consecutive measurements is small, we can in fact conclude that the information that the robot has moved after it was at $x(k-1)$ does not provide any information about $x(k-1)$. Note that this is not true in general. Suppose the environment of the robot consists of two rooms, a small and a large room, and that there is no door between these two rooms. Furthermore suppose the robot moved a distance that is larger than the diameter of the small room. After that movement the probability that the robot is in the larger room must exceed the probability that the robot is in the smaller room. If, however, the time intervals between consecutive measurements are small, each movement can only represent a small distance and the fact that the robot has moved a few inches away from its current location $x(k-1)$ carries almost no information about $x(k-1)$ given we do not know $x(k)$. Under this assumption we therefore can conclude that $u(k-1)$ does not provide information about $x(k-1)$ if we have no information about $x(k)$. Thus, we assume that $x(k-1)$ is independent of $u(k-1)$ in the term $P(x(k-1) \mid u(0:k-1), y(1:k-1))$. Thus we obtain

$$\begin{aligned} &P(x(k-1) \mid u(0:k-1), y(1:k-1)) \\ &\quad = P(x(k-1) \mid u(0:k-2), y(1:k-1)) \end{aligned} \tag{9.13}$$

and simplify (9.12) to

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k-1)) \\ &\quad = \sum_{x(k-1) \in X} (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad \cdot P(x(k-1) \mid u(0:k-2), y(1:k-1))). \end{aligned} \tag{9.14}$$

If we now substitute this result into (9.9) we obtain

$$\begin{aligned} &P(x(k) \mid u(0:k-1), y(1:k)) \\ &\quad = \eta(k)\ P(y(k) \mid x(k)) \sum_{x(k-1) \in X} (P(x(k) \mid u(k-1), x(k-1)) \\ &\qquad P(x(k-1) \mid u(0:k-2), y(1:k-1))) \end{aligned} \tag{9.15}$$

which directly corresponds to the (9.1).

9.1.4 Representations of the Posterior

As mentioned above, the probabilistic formulation leaves open how the posterior is represented. In principle, there are various ways to represent the posterior. Mathematically speaking, P is a function $P : X \to \mathbb{R}$. When X is continuous (or infinite), P lives

in an infinitely dimensional space. Of course it is impossible to arbitrarily represent an infinitely dimensional map. Thus we have to be content with a finite approximation. Throughout this section we discuss three different approaches: Kalman filters, discrete approximations, and particle filters.

Kalman Filters

The previous chapter covered a common approach to the representation of the belief $P(x(k) \mid u(0:k-1), y(1:k))$ as Extended Kalman filters (EKFs) [215, 390]. In this case, the posterior is represented using a unimodal Gaussian distribution. Many successful applications of Kalman filters for mobile robot localization have been demonstrated [28, 179, 276, 371]. One advantage of Kalman filtering is that it can be implemented quite efficiently and that it works well in high-dimensional state spaces. Additionally, Kalman filters provide a floating-point resolution and in this way allow highly accurate estimates.

Unfortunately, Kalman filters are only optimal for systems whose behavior is governed by the linear equations given by (8.1) and (8.2). Since Kalman filters use Gaussian distributions, they cannot appropriately represent beliefs that correspond to ambiguous situations as they appear, e.g., in the context of global localization. As a result, localization approaches using Kalman filters typically require that the starting location of the robot is known or that unique landmarks are given so that there is no data association problem. To overcome the limitations of Kalman filters, recent extensions of this approach have been developed. For example, Jensfeld and Christensen [209] use a mixture of Gaussians to represent the belief about the location of the robot. They also present techniques to update this mixture based on sensory input and robot motions. In this chapter, we consider two alternative nonparametric state representations—discrete grids and samples—to bypass the Gaussian assumption.

Discrete Approximations

An alternative form to represent $P(x(k))$ is to use a discrete approximation of the configuration space. Independent of the structure of the discretization all approaches store in each element of their discrete structure the probability that the robot is at the location that corresponds to this element. In practice, one mainly finds topological and geometric discretizations. In the first case, the configuration space is separated according to the topological structure of the environment. Many systems that exploit topological structures use individual states for junctions, doorways and rooms and four possible headings. Several systems [213, 339, 386] follow this approach and perform probabilistic localization for landmark-based corridor

navigation. Choset and Nagatani exploit the topology of the generalized Voronoi diagram (GVD) [108], described in chapter 5. The advantage of a topological representation lies in its compactness, because only a limited number of states need to be considered. Its disadvantage, on the other hand, is limited accuracy with respect to position and orientation. To achieve more accurate estimates, Burgard, Fox, and coworkers [83, 158] use a fine-grained grid to represent the posterior. Throughout this section we will give a detailed description of this grid-based technique.

If we assume that the configuration of the robot is $SE(2)$ and thus a configuration is represented by a three-dimensional random vector consisting of the (x_r, y_r)-position and the orientation θ_r of the vehicle, our grid needs to be three-dimensional. Whereas the first two dimensions are used for the position of the vehicle, the third dimension is used for its orientation. Figure 9.2 shows the structure of the grid for this kind of representation. In practical applications of this technique, spatial resolution of 10 to 30 cm and an angular resolutions of 2 to 10 degrees turned out to be sufficient for robust and accurate localization of mobile robots [81, 177].

To compute the posterior $P(x(k) \mid u(0:k-1), y(1:k))$ represented by a grid we follow the procedure summarized in Algorithm 16. Given $u(0:k-1)$, $y(1:k)$ and an initial belief $P(x(0))$, we carry out k loops. In the every round i we integrate $u(i-1)$ and $y(i)$. The first step (i.e., the prediction step) incorporates the movement $u(i-1)$, which means that we have to recompute the grid according to the motion model $P(x \mid u(i-1), x')$. In principle, this involves integrating over all possible prior states of the robot, which would result in an $O(N^2)$ complexity, where N is the number of states represented by the grid. One way to reduce the complexity of this operation is to

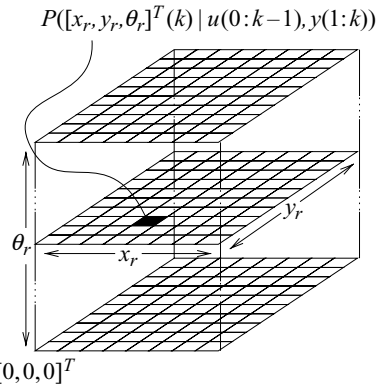

Figure 9.2 Grid-based representation of the state space.

Algorithm 16 Probabilistic localization for discrete state spaces

Input: Sequence of measurements $y(1:k)$ and movements $u(0:k-1)$ and initial belief $P(x(0))$

Output: A posterior $P(x(k) \mid u(0:k-1), y(1:k))$ about the configuration of the robot at time step k

1: $P(x) \leftarrow P(x(0))$
2: **for** $i \leftarrow 1$ to k **do**
3: **for all** states $x \in X$ **do**
4: $P'(x) \leftarrow \sum_{x' \in X} P(x \mid u(i-1), x') \cdot P(x')$
5: **end for**
6: $\eta \leftarrow 0$
7: **for all** states $x \in X$ **do**
8: $P(x) \leftarrow P(y(i) \mid x) \cdot P'(x)$
9: $\eta \leftarrow \eta + P(x)$
10: **end for**
11: **for all** states $x \in X$ **do**
12: $P(x) \leftarrow P(x)/\eta$
13: **end for**
14: **end for**

limit the number of predecessor states summed over. In a successful implementation of the grid-based representation Fox, Burgard and Thrun [158] applied the following approach to approximate the integration over all potential previous states: First, all grid cells are shifted according to the motion $u(i-1)$ carried out by the robot, and then the whole grid is convolved using a bounded Gaussian kernel that corresponds to the uncertainty of $u(i-1)$. Whenever the robot has moved, we can easily compute for every (x, y)-plane of the grid the offsets Δx and Δy by which each cell has to be shifted according to $u(i-1)$ (and assuming there are no odometry errors). Please note that Δx and Δy both depend on the angle θ that the corresponding plane in the grid represents. The convolution operation can be carried out efficiently using a separable kernel. This involves convolving independently over the individual dimensions of the grid. To realize this, one usually introduces a one-dimensional array P' that stores the intermediate results of this computation. In the case of the x-dimension we proceed as follows for all x:

$$\begin{aligned} P'((x, y, \theta)) &= 0.5 \cdot P((x, y, \theta)) \\ &\quad + 0.25 \cdot (P((x-1, y, \theta)) + P((x+1, y, \theta))) \end{aligned} \tag{9.16}$$

Special care has to be taken at the borders of the grid. In this case only one neighboring cell is given and one chooses the coefficients $\frac{2}{3}$ for the cell itself and $\frac{1}{3}$ for the neighbor cell ($x+1$ or $x-1$ depending on where one is in the grid). Whenever $P'((x, y, \theta))$ has been computed for all x and a given pair of y and θ, the results are then stored back in the original cells. Similarly we proceed with all y and θ. To correctly model the uncertainty introduced by the motion $u(i-1)$ the convolution process can be repeated appropriately. The second step (i.e., the update step) of Algorithm 16 integrates the observation $y(i)$ into the grid. To achieve this, we simply multiply every grid cell by the likelihood of the observation $y(i)$, given the robot has the configuration corresponding to that particular cell. Afterward the whole grid is normalized.

Algorithm 16 can also be used for incremental filtering. If the initial belief is set to the output obtained from the preceeding application of the algorithm and if all measurements and movements since this point in time are given as input, the algorithm incrementally computes the corresponding posterior. Please also note that Algorithm 16 can easily be extended to situations in which the movements and the environment measurements do not arrive in an alternating and fixed scheme.

One important aspect of all state estimation procedures is the extraction of relevant statistics such as the mean and the mode. These parameters are important whenever the robot has to generate actions based on the current belief about its state. For example, this can be the next motion command to enter a specific room. Both the mode and the mean can be determined efficiently given a grid-based approximation. The x- and y-coordinates of the mean ($\tilde{x}$ and $\tilde{y}$) can be computed by computing the weighted sums $i = 1, \ldots, N$ over all cells of the grid. To compute the angle mean we use the following equation:

$$\hat{\phi} = \operatorname{atan2}\left(\sum_{i=0}^{N-1} P(i) \cdot \sin\phi(i), \sum_{i=0}^{N-1} P(i) \cdot \cos\phi(i)\right) \tag{9.17}$$

Unfortunately, the mean has the disadvantage that the resulting values can lack any useful meaning, especially in the context of multimodal distributions. For example, the mean of a bimodal distribution might lie within an obstacle so that no meaningful commands can be generated. An alternative statistic is the mode of the distribution which, given a grid-based approximation, can be computed efficiently by a simple maximum operation. Compared to the mean, the mode has the advantage that it generally corresponds to a possible location of the vehicle. However, the locations of subsequent modes can differ largely so that the estimates are not as continuous as if we choose the mean. Whereas the mean automatically yields estimates at subgrid-resolution accuracy, we can obtain the same for the mode by averaging over a small region around the cell containing the maximum probability [82].

To illustrate an application example of a grid-based representation, consider the map and the data set depicted in the left image of figure 9.3. The map, which was generated using the system described by Buhmann and coworkers [73], corresponds to the environment of the AAAI '94 mobile robot competition. The size of this environment is 31 by 22 m. The right image of the same figure depicts the path of the B21 robot Rhino [415] along with measurements of the twenty four ultrasound sensors obtained as the robot moved through the competition arena. Here we use this sensor information to globally localize the robot from scratch. The time required to process this data on a 400 MHz Pentium II is 80 seconds, using a position probability grid with a spatial resolution of 15 cm and an angular resolution of 3 degrees.

The right image of figure 9.3 also marks the points in time when the robot perceived the fifth (A), eighteenth (B), and twenty-fourth (C) sensor sweep. The posteriors during global localization at these three points in time are illustrated in figure 9.4. The figures show the belief of the robot projected onto the (x, y)-plane by plotting

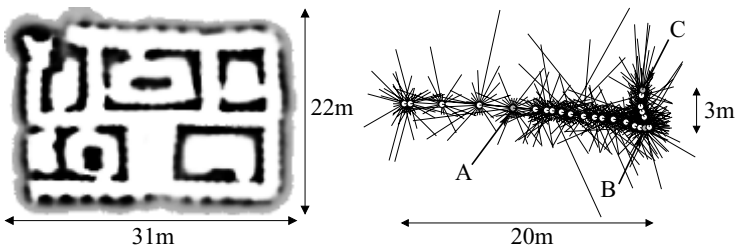

Figure 9.3 Occupancy grid map of the 1994 AAAI mobile robot competition arena (left) and data set recorded in this (right). It includes the odometry information and the ultrasound measurements. Point *A* is after five steps, *B* is after eighteen, and *C* is after twenty-four.

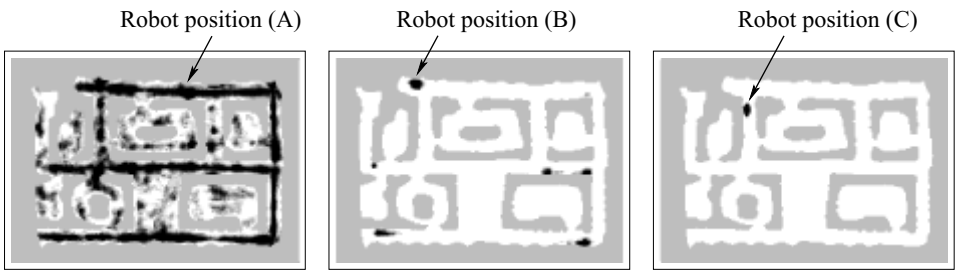

Figure 9.4 Density plots after incorporating five, eighteen, and twenty-four sonar scans (the darker locations are more likely).

for each (x, y)-position the maximum probability over all possible orientations. More likely locations are darker and, for illustration purposes, the left and middle images use a logarithmic scale in intensity. The leftmost image of figure 9.4 shows the belief state after integrating five sensor sweeps (i.e. when the robot is at step A on its path). At this point in time, all the robot knows is that it is likely in one of the corridors of the environment. After integrating eighteen sweeps of the ultrasound sensors (at step B) the robot is almost certain that it is at the end of a corridor (see center image of figure 9.4). After incorporating twenty-four scans (step C) the robot has determined its location uniquely. This is represented by the unique peak containing 99% of the whole probability mass (see rightmost image of figure 9.4).

Although the grid-based approach has the advantage that it provides a well-understood approximation of the true distribution and that important statistics such as the mean and the mode can be easily assessed, it has certain disadvantages. First, the number of grid cells grows exponentially in the number of dimensions and therefore limits the application of this approach to low-dimensional state spaces. Additionally, the approach uses a rigid grid. If the whole probability mass is concentrated on a unique peak, most of the states in the grid are useless and approaches that focus the processing time on regions of high likelihood are preferable. One method to dynamically adapt the number of states that have to be updated is the selective updating scheme [158]. Burgard, Derr, Fox, and Cremers [82] use a tree structure and store only cells whose probability exceeds a certain threshold. In this way, memory and computational requirements can be adapted to the complexity of the posterior.

Particle Filters

An alternative and efficient way of representing and maintaining probability densities is the particle filter. The key idea of particle filters is to represent the posterior by a set $\mathcal{M}$ of N samples. Each sample consists of a pair (x, ω) containing a state vector x of the underlying system and a weighting factor ω, i.e., $\mathcal{M} = (X, [0, 1])^N$. The latter is used to store the importance of the corresponding particle. The posterior is represented by the distribution of the samples and their importance factors. In the past a variety of different particle filter algorithms have been developed and many variants have been applied with great success to various application domains [97,125,138,157,167,201,218]. Algorithm 17 describes a particle filter algorithm that uses *sequential importance sampling with resampling* [29] to implement the update step. This algorithm follows a survival of the fittest scheme. Whenever a new measurement $y(k)$ arrives, the weight ω of a particle (x, ω) is computed as the likelihood $p(y(k) \mid x)$ of this observation given the system is in state x. After computing

Algorithm 17 Probabilistic localization using a particle filter

Input: Sequence of measurements $y(1:k)$ and movements $u(0:k-1)$ and set $\mathcal{M}$ of N samples (x_j, ω_j) corresponding to the initial belief $P(x)$

Output: A posterior $P(x(k) \mid u(0:k-1), y(1:k))$ about the configuration of the robot at time step k represented by $\mathcal{M}$.

```
1:  for i ← 1 to k do
2:    for j ← 1 to N do
3:      compute a new state x by sampling according to P(x | u(i − 1), x_j).
4:      x_j ← x
5:    end for
6:    η ← 0
7:    for j ← 1 to N do
8:      w_j = P(y(i) | x_j)
9:      η = η + w_j
10:   end for
11:   for j ← 1 to N do
12:     w_j = η^{-1} · w_j
13:   end for
14:   M = resample(M)
15: end for
```

the weights, a so-called resampling procedure is applied. We draw N samples with replacement from $\mathcal{M}$ such that each sample in $\mathcal{M}$ is selected with a probability that is proportional to its weight ω. Accordingly, samples with greater weights survive with higher likelihood than samples with values of small importance. In principle, there are many ways of achieving this. One popular approach (see also [29]) is described by algorithm 18. In this algorithm, the procedure *rand(I)* draws a random value from the interval I according to a uniform distribution. The major advantage of this algorithm is that the whole resampling process is carried out in $O(N)$ steps. One alternative technique is the one used by Isard and Blake [201]. This approach relies on binary search to select a sample and thus requires $O(N \log N)$ steps.

We also need to describe the prediction step that we use to incorporate the motions of the robot into the sample set. Throughout this chapter we assume that incremental motions of a robot between two configurations x_1 and x_2 are encoded by the three parameters α, β, and d (see figure 9.5). Here α is an initial rotation in x_1 toward x_2, d is the distance to be traveled from x_1 to x_2, and β is the final rotation carried out at the location x_2 to reach the orientation of the robot in x_2. Since the motions carried

Algorithm 18 The procedure *resample*($\mathcal{M}$)

Input: Set $\mathcal{M}$ of N samples
Output: Set $\mathcal{M}'$ of N samples obtained by importance resampling from $\mathcal{M}$

1: $\mathcal{M}' \leftarrow \emptyset$
2: $\Delta \leftarrow rand((0; N^{-1}])$
3: $c \leftarrow \omega_0$
4: $i \leftarrow 0$
5: **for** $j \leftarrow 0$ to $N-1$ **do**
6: $\quad u \leftarrow \Delta + j \cdot N^{-1}$
7: $\quad$ **while** $u > c$ **do**
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad c \leftarrow c + \omega_i$
10: $\quad$ **end while**
11: $\quad \mathcal{M}' \leftarrow \mathcal{M}' \cup \{(x_i, N^{-1})\}$
12: **end for**

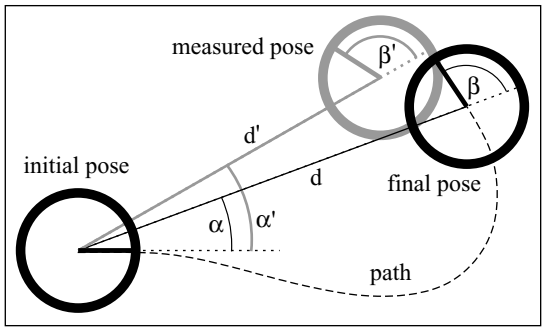

Figure 9.5 The parameters α, β, and d specifying any incremental motion of a robot in the (x, y, θ)-space.

out by the robot are not deterministic, we need to cope with potential errors when we compute new locations for samples. We proceed as follows: Whenever we compute the new configuration for a sample after a movement $u(i-1)$, we incorporate the possible deviations from the values of α, β, and d. Throughout this section, we assume Gaussian noise in these values and compute the new location of a sample according to values α', β', and d' that deviate from the measured values α, β, and d according to Gaussian distributions. If we denote the robot state as $x = [x_r, y_r, \theta_r]$

and the ith particle by x^i, then the motion model is implemented by assigning

$$x^i := x^i + \begin{bmatrix} x_r^i + d' \cos(\theta_r^i + \alpha') \\ y_r^i + d' \sin(\theta_r^i + \alpha') \\ \theta_r^i + \alpha' + \beta' \end{bmatrix} \tag{9.18}$$

for each particle in the collection, where

$$\alpha' = \alpha + \alpha \cdot \text{norm}(\sigma_1) + d \cdot \text{norm}(\sigma_2) \tag{9.19}$$

$$\beta' = \beta + \beta \cdot \text{norm}(\sigma_3) + d \cdot \text{norm}(\sigma_4) \tag{9.20}$$

$$d' = d + d \cdot \text{norm}(\sigma_5) + (\alpha + \beta) \cdot \text{norm}(\sigma_6). \tag{9.21}$$

Here $\text{norm}(\sigma)$ is a random number generator that outputs random numbers according to a normal distribution with mean 0 and standard deviation σ. The standard deviations σ_i are parameters that describe the influence of the translation d and the rotations α and β on the potential errors. In this model the errors in all three values depend on the rotations and the translation carried out. Note that the σ_i can be learned by generating a statistic about typical deviations of the actual movements from the values α, β, and d.

Figure 9.6 illustrates an application of this motion model to a sample set in which all samples are concentrated in a single state. The line depicts the path taken by the robot and the sample sets illustrate the belief about the robot's configurations at

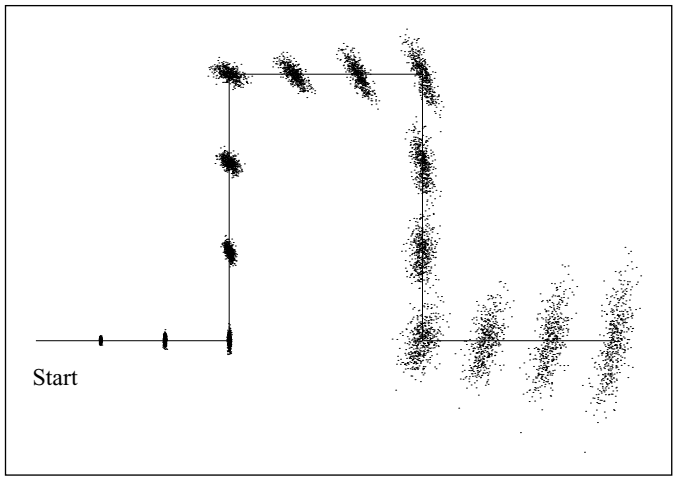

Figure 9.6 Sample-based approximation of the belief of the robot after repeatedly executing motion commands. In this example, the robot did not perceive the environment while it was updating the sample set.

certain points in time. In this particular example we incorporated no observations of the environment into the sample set. As can be seen from the figure, the robot's uncertainty grows indefinitely while it is moving, and the overall distribution is slightly bent.

Note that the motion model described above does not incorporate any information about the environment. Accordingly, samples might end up inside obstacles in the map of the environment. One advantage of sample-based approaches, however, lies in the fact that such environmental information can easily be incorporated. To avoid that samples move through obstacles we can simply reject such values for α', β', and d'.

To extract the mean of a posterior represented by N samples, we can proceed in a similar way as for the grid-based representation. We simply average over all samples of the distribution. In the case that not all importance factors are equal we use the normalized importance factors as weighting factors. When computing the mode, we distinguish two different situations. If we compute the mode just before the resampling step, we can simply select that sample with the highest importance factor, which requires $O(N)$ steps. After resampling, however, the mode cannot be computed as easily, because the actual form of the posterior is only encoded in the density of the samples. One popular approach to approximate the mode is to use kd-trees [44], which, however, requires $O(N \log N)$ steps. Alternatively, one can compute a histogram based on a coarse discretization of the state space. In this case, the space requirements are similar to the grid-based approach, but the mode can be extracted in $O(N)$ steps.

Figure 9.7 shows a particle filter in action. This example is based on the ultrasound and odometry data obtained while the robot traveled along the path depicted in figure 8.1 in chapter 8. To achieve global localization we initialized the filter by selecting the initial set of particles from a uniform distribution over the free space in the environment (see left image of figure 9.7). After incorporating ten ultrasound

Figure 9.7 Global localization using a particle filter with 10,000 samples. The left image shows the initial distribution. The middle image shows the distribution after incorporating ten ultrasound beams. The right image shows a typical situation (here after 65 steps) when the location of the robot has been identified.

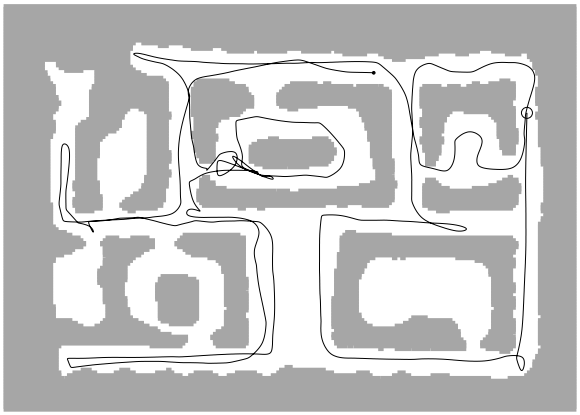

Figure 9.8 Trajectory of the robot obtained by applying a particle filter for tracking the configuration of the robot using the data gathered along the path depicted in figure 8.1 in chapter 8.

measurements we obtain the particle set depicted in the middle image of figure 9.7. After incorporating 65 measurements, the particle filter has converged to configurations close to the true location of the robot. The resulting density is depicted in the right image of figure 9.7.

Figure 9.8 shows the path of a robot as it is estimated by a particle filter. Here the particle filter was initialized using a Gaussian distribution and therefore was just tracking the location of the robot. Again, the odometry data used as input are shown in figure 8.1. Additionally, the filter used the data of the 24 ultrasound sensors. As can be seen, the particle filter can robustly track the position of the robot, although ultrasound measurements are noisy and there are larger errors in odometry.

As the global localization example illustrates, particle filters typically converge to the most likely estimate, i.e., after a certain period of time all of the particles usually cluster around the true configuration of the system. While this is desired in most situations, it also can be disadvantageous. This is especially true if a failure occurs that is not modeled in the motion model. One typical scenario in which such unpredicted localization errors frequently occur is the RoboCup environment [237]. There are certain conditions under which a referee removes a player from the soccer field. After a short period of time the player is then placed back onto the field. The problem that has to solved by the robot in such a case is usually denoted as the "kidnapped robot problem" [145]. To deal with such situations, or more generally, with situations in which the estimation process fails, the robot requires techniques to detect localization failures and to initiate a global localization. One approach is to

modify the motion model and to choose random configurations for a certain fraction of the samples [156]. Alternatively, one can monitor the average observation likelihood $\tilde{p} = N^{-1} \cdot \sum_{j=1}^{N} P(y(i) \mid x_j)$ of all samples. For example, Burgard et al. [82] restart a global localization if this value falls below a certain threshold. Lenser and Veloso [275] adjust the number of samples with randomly chosen locations according to the value of $\tilde{p}$. Gutmann and Fox [176] additionally smooth $\tilde{p}$ to be more robust against short-term changes of $\tilde{p}$.

9.1.5 Sensor Models

One of the crucial aspects of probabilistic localization is how the likelihood of the robot's sensor measurements is computed. In particular, we are interested in the quantity $P(y \mid x)$, which represents the likelihood of measuring y given x is the location of the system. Throughout this chapter, we denote the way in which we compute this quantity as the sensor model. Obviously, a good sensor model largely depends on the type of sensor that is used for localization. Additionally, it also may depend on the environment. For example, it might exploit particular features of the environment, such as landmarks. Finally, it also depends on the way the environment is represented, i.e., on the type of the map.

In this subsection we describe a sensor model that captures several of the physical properties of frequently used proximity sensors such as ultrasound or laser range scanners. To motivate this sensor model let us first investigate a typical scan obtained with the 24 ultrasound sensors of a B21 robot. One such scan is shown in figure 9.9. In this figure the objects in the environment are shown in light gray. The dark lines indicate the central axis of 24 ultrasound beams as they are obtained at the corresponding location in this environment. As can be seen from the figure, most of the measurements are quite accurate. For example, the beams 0, 2, 3, 4, 10, 13, and 15 quite accurately correspond to the distance to the nearest obstacle in the measurement direction. Other beams, such as 1, 12, and 14, are shorter than the distance to the nearest obstacle. In this particular situation, the measurement 1 resulted from a crosstalk: the sensor received a sound signal emitted by another sensor [61]. The other two short measurements (12 and 14) were caused by objects not contained in the map. Whereas beam 12 was reflected by a person entering the room, the cone of beam 14 was echoed by a refrigerator installed in the niche. Furthermore, some of the measurements, such as 18, 19, and 20, appear to be quite random. They apparently pass through a bookshelf and appear to be echoed by an unmodeled object behind it. Finally, the sensors 6, 8, and 23 never received an echo and therefore report a maximum range reading.

The sensor model that we describe in the remainder of this subsection is designed to capture the noise and error characteristics of many active range sensors. It can model

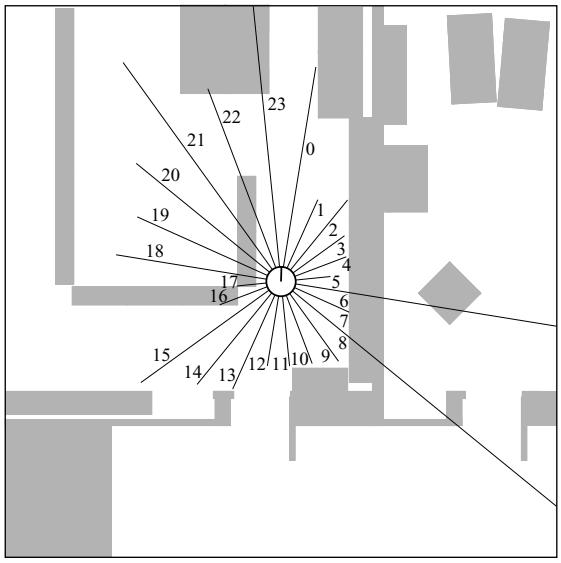

Figure 9.9 Ultrasound scan perceived with a B21 robot in an office environment.

the accuracy of the sensor whenever the beam hits the nearest object in the direction of the measurement. Additionally, it represents random measurements. It furthermore provides means to model objects not contained in the map and to represent the effects of crosstalk between different sensors. Finally, it incorporates a technique to deal with detection errors in which the sensor reports a maximum range measurement. The model has been applied successfully in the past for mobile robot localization with proximity sensors such as ultrasound sensors and laser range finders. In 1997 and 1998 [81] the mobile robots Rhino and Minerva operated several weeks in populated museum environments using the sensor model described here for localization with laser range scanners.

Throughout this subsection we assume that range sensors have a limited numerical resolution, i.e., the information they provide is discrete. Accordingly, we consider a discrete set of distances $d_0, \ldots, d_{n-1}$ where d_{n-1} corresponds to the maximum distance that can be measured. We also assume that the size of the ranges $\Delta = \Delta_{i+1} = d_{i+1} - d_i$ is the same for all i. In principle, the distribution $P(y \mid x)$ that y is observed given the state of the system is x can be specified by a histogram that stores in each of its n bins the likelihood that y is d_i, $i = 0, 1, \ldots n-1$. Obviously, storing an individual histogram for sufficiently large number of potential states would consume too much space.

The key idea of the model that we describe here is to compute $P(y \mid x)$ based on the distance $d(x)$ to the closest obstacle in the map within the perceptual field of the sensor. If the environment is represented geometrically, i.e., by an evidence grid or by geometric primitives such as polygons or lines, the expected distance $d(x)$ can be computed efficiently using ray-casting. It is natural to assume that

$$P(y \mid x) = P(y \mid d(x)). \tag{9.22}$$

Accordingly, it suffices to determine the expected distance d for the given measuring direction at the current location of the vehicle and then compute $P(y \mid d)$.

The quantity $P(y \mid d)$ is calculated as follows. According to the different situations identified in the scan depicted in figure 9.9, we distinguish the following four situations:

1. *The nearest object in the direction of the beam is detected.* The actual measurement depends on the accuracy of the underlying sensor. Typically, the likelihood of the measurement y is then well-approximated by a Gaussian distribution $\mathcal{N}(y, d, \sigma)$, where d is the true distance to the object and the variance σ depends on the accuracy of the sensor; it is higher for ultrasound sensors than for laser range scanners.

2. *An object not contained in the map reflects the beam, or there is crosstalk.* The sensor will report a distance that is shorter than the expected distance. In our model, we represent this by an exponential distribution, e.g., $\lambda e^{-\lambda y}$.

3. *The sensor produces a random measurement.* As mentioned above, there are situations in which the sensor provides a random measurement that cannot be explained given the current map. We model these types of measurements by a uniform distribution over the possible distances reported by the sensor, represented by a constant γ.

4. *The sensor reports a maximum range reading.* In some situations, time-of-flight sensors such as ultrasounds or laser range scanners, or intensity-based sensors such as those based on infrared light fail to detect the beam reflected by an object. If no random measurement is obtained and if no crosstalk happens, the sensor may report a maximum range measurement. The likelihood of this event is represented by a constant δ.

Since we do not know which situation is given, our distribution needs to represent all the different cases. Accordingly, the distribution $P(y \mid d)$ is computed based on a mixture of the four different densities that correspond to the individual situations. Suppose d_i is the expected distance in a particular measurement direction at a given

position in the environment. Then we determine a complete histogram h_i containing in each bin $h_{i,j}$ the likelihood of each possible measurement d_j the robot can obtain. If $j < n-1$, i.e., the actual measurement is not a maximum range measurement, $h_{i,j}$ is computed based on a mixture of the three densities representing the situations 1, 2, and 3 described above. If, however, $j = n-1$, then we have a single value representing the likelihood of maximum range measurements.

$$h_{i,j} = \begin{cases} \int_{d_j - \frac{\Delta}{2}}^{d_j + \frac{\Delta}{2}} \alpha_i \cdot \mathcal{N}(y, d_i, \sigma_i) + \beta_i \cdot \lambda_i e^{-\lambda_i y} + \gamma_i \; dy & \text{if } j < n-1 \\ \delta_i & \text{if } j = n-1 \end{cases} \tag{9.23}$$

The values of the parameters σ_i, α_i, β_i, γ_i and δ_i have to be chosen appropriately such that each h_{ij} reflects the correct likelihood. Thereby, we have to consider the constraint that

$$\sum_{j=0}^{n-1} h_{i,j} = 1 \tag{9.24}$$

for each histogram h_i. A typical approach to determine the optimal values for the different parameters of each histogram is log-likelihood maximization. Given a set of actual measurements for a given expected distance d_i this approach seeks to determine those values for the parameters that maximize the sum of the log likelihoods $\log P(d \mid d_i)$ over all measurements in the data set.

Figures 9.10 and 9.11 show the resulting maximum-likelihood approximations for two different expected distances. Whereas figure 9.10 plots the histograms for ultrasound data, figure 9.11 plots the histograms obtained for laser range data. In all plots, the data are shown as boxes and the approximation is shown as dots. As can

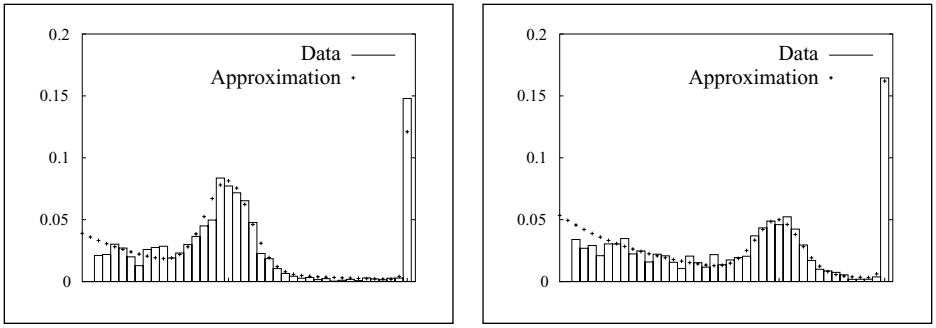

Figure 9.10 Histograms (data and maximum-likelihood approximation) for data sets obtained with ultrasound sensors and for two different expected distances.

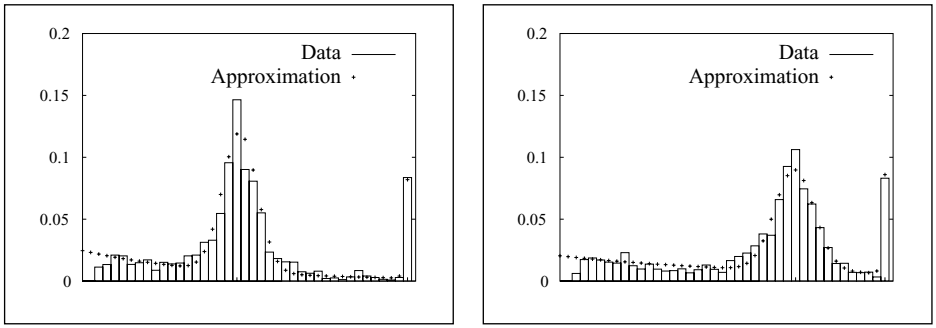

Figure 9.11 Histograms (data and maximum-likelihood approximation) for data sets obtained with a laser range finder and for two different expected distances.

be seen, our model is quite accurate and reflects well the properties of the real data. It is furthermore obvious that the laser range finder yields much more accurate data than the ultrasound sensor. This is represented by the fact that the Gaussians have a lower variance in both histograms. Please also note that the variances of the Gaussians do not depend solely on the accuracy of the sensor. They also encode the uncertainty of the map. For example, if dynamic objects such as chairs or tables are slightly moved, the error in the measurements increases, which results in a higher uncertainty of the sensor model.

Range scans obtained with laser range scanners typically consist of multiple measurements. Some robots are also equipped with arrays of ultrasound sensors which provide several measurements $y_1, y_2, \ldots y_m$ at a time. In practice it is often assumed that these measurements are independent, i.e, that

$$P(y_1, y_2, \ldots, y_m \mid x) = \prod_{i=1}^{I} P(y_i \mid x). \tag{9.25}$$

In general, however, this assumption is not always justified, e.g., if the environment contains objects that are not included in the map. In such a situation, the knowledge that one measurement is shorter than expected raises the probability that a neighboring beam of the same scan that intercepts a region close to the first measurement is also shorter. A popular solution to this problem is to use only a subset of all beams of a scan. For example, Fox et al. [158] typically used only 60 beams of the 181 beams of a SICK PLS laser range scan.

Finally, we want to describe some aspects that might be important when implementing this model. The first disadvantage of this model is that one has to perform a

ray-casting operation for every potential state of the system. In a grid-based representation of the state space this involves a ray-casting operation for every cell of the grid. If a sample-based representation is used, the same operation has to be performed for every sample in the sample set. One approach to reduce the computation time is to precompute all expected distances and to store them in a large lookup table. Given an appropriate discretization of the state space, the individual entries of this table can be accessed in constant time. If, furthermore, one limits the resolution of the range data such that each beam can have no more than 256 values, only one byte is needed for each entry of the table. In this case, also, the number of histograms for the computation of $P(y \mid d)$ is limited. We only need 256 histograms with 256 bins each.

A further disadvantage of this approach is that the likelihood function sometimes lacks smoothness with respect to slight changes of the locations of the robot. For example, consider a situation in which a laser range finder points into a doorway. If we move the robot slightly, the beam might hit the adjacent wall. Alternatively, consider a beam that hits a wall at a small angle. Slight changes in the orientation of the robot in this case have a high influence on the measured distance. In contrast to that, if the beam is almost perpendicular to a wall, slight changes of the orientation of the robot have almost no influence on the measured distance. One way to solve this problem is to also consider the variance of the expected measurement. This variance can also be computed beforehand by integrating over the local neighborhood of the state x. Given an appropriate discretization of the variances, the histograms then have to be learned for each pair of expected distances and variance. Both techniques, the compact representation of expected distances and the integration of the variance of the expected distance with respect to slight changes in the location of the robot, have been used successfully in Rhino and Minerva [81, 158, 414].

Note that several alternative models have been proposed in the past. The goal of all these models is to provide robust and accurate estimates of the location of the robot. For example, Moravec [324] and Elfes [143], who introduced occupancy grid maps, also presented a probabilistic technique to compute the likelihood of ultrasound measurements given such a map and the position of the robot. Yamauchi and Langley [430] compared local maps built from the most recent measurements with a global map. The sensor models proposed by Konolige [245] and Thrun [412] are more efficient than the model presented here since they avoid the ray-casting operation and only consider the endpoint of each beam. Finally, Simmons and Koenig [386] extracted doorways and corridor junction types out of local grid maps and compared this information to landmarks stored in a topological representation of the environment.

9.2 Mapping

In chapter 8 we learned how to use a Kalman filter for acquiring a map of the environment. The assumption there was that the robot can identify landmarks in the environment and that the posterior about the location of the robot and the landmarks can be represented by a Gaussian distribution. In this section we consider probabilistic forms of mapping that—similarly to probabilistic localization—allow representation of arbitrary posteriors about the state of the environment and the location of the robot during mapping.

To map an environment, a robot has to cope with two types of sensor noise: noise in perception (e.g., range measurements), and noise in odometry (e.g., wheel encoders). Because of the latter, the problem of mapping creates an inherent localization problem. The mobile robot mapping problem is therefore often referred to as the *concurrent mapping and localization problem (CML)* [277], or as the *simultaneous localization and mapping problem* [99, 128] (see also chapter 8). As in chapter 8 we will use the acronym SLAM when referring to the latter. In fact, errors in odometry render the errors of individual features in the map dependent even if the measurement noise is independent, which suggests that SLAM is a high-dimensional statistical estimation problem, often with tens of thousands of dimensions. In this chapter we approach this problem in two steps. First we concentrate on the question of how to build maps given the location of the robot is known. Afterward we relax this assumption and describe a recently developed technique for SLAM.

9.2.1 Mapping with Known Locations of the Robot

A very popular, probabilistic approach to represent the environment is the so-called occupancy probability grid pioneered by Elfes and Moravec in the 80s [325]. Occupancy probability grids are approximative. Each cell m_l of such a two-dimensional grid m stores the probability $P(m_l \mid x(1:k),\, y(1:k))$ that the place in the environment corresponding to m_l is occupied given the observations $y(1:k) = y(1), \ldots, y(k)$ and all locations of the robot $x(1:k) = x(1), \ldots, x(k)$ at the corresponding points in time. Because of their probabilistic nature, occupancy probability grids can be updated easily based on sensory input.

Occupancy probability grids seek to find the map m that maximizes $P(m \mid x(1:k),\, y(1:k))$. If we apply Bayes rule using $x(1:k)$ and $y(1:k-1)$ as background knowledge, we obtain

$$\begin{aligned} &P(m \mid x(1:k),\, y(1:k)) \\ &\quad = \frac{P(y(k) \mid m,\, x(1:k),\, y(1:k-1))\; P(m \mid x(1:k),\, y(1:k-1))}{P(y(k) \mid x(1:k),\, y(1:k-1))}. \end{aligned} \tag{9.26}$$

If we assume that $y(k)$ is independent from $x(1:k-1)$ and $y(1:k-1)$ given we know m, then the right side of this equation can be simplified to

$$P(m \mid x(1:k), y(1:k)) = \frac{P(y(k) \mid m, x(k))\ P(m \mid x(1:k), y(1:k-1))}{P(y(k) \mid x(1:k), y(1:k-1))}. \tag{9.27}$$

We now again apply Bayes rule to determine

$$P(y(k) \mid m, x(k)) = \frac{P(m \mid x(k), y(k))\ P(y(k) \mid x(k))}{P(m \mid x(k))}. \tag{9.28}$$

If we insert (9.28) into (9.27) and since $x(k)$ does not carry any information about m if there is no observation $y(k)$, we obtain

$$P(m \mid x(1:k), y(1:k)) = \frac{P(m \mid x(k), y(k))\ P(y(k) \mid x(k))\ P(m \mid x(1:k-1), y(1:k-1))}{P(m)\ P(y(k) \mid x(1:k), y(1:k-1))}. \tag{9.29}$$

If we exploit the fact that each m_l is a binary variable, we derive the following equation for the posterior probability that all cells of m are free in an analogous way.

$$P(\neg m \mid x(1:k), y(1:k)) = \frac{P(\neg m \mid x(k), y(k))\ P(y(k) \mid x(k))\ P(\neg m \mid x(1:k-1), y(1:k-1))}{P(\neg m)\ P(y(k) \mid x(1:k), y(1:k-1))}, \tag{9.30}$$

where $\neg m$ denotes the complement of m. By dividing (9.29) by (9.30), we obtain

$$\frac{P(m \mid x(1:k), y(1:k))}{P(\neg m \mid x(1:k), y(1:k))} = \frac{P(m \mid x(k), y(k))\ P(\neg m)\ P(m \mid x(1:k-1), y(1:k-1))}{P(\neg m \mid x(k), y(k))\ P(m)\ P(\neg m \mid x(1:k-1), y(1:k-1))}. \tag{9.31}$$

Finally, we use the fact that $P(\neg A) = 1 - P(A)$ and obtain the following equation:

$$\frac{P(m \mid x(1:k), y(1:k))}{1 - P(m \mid x(1:k), y(1:k))} = \frac{P(m \mid x(k), y(k))}{1 - P(m \mid x(k), y(k))} \frac{1 - P(m)}{P(m)} \frac{P(m \mid x(1:k-1), y(1:k-1))}{1 - P(m \mid x(1:k-1), y(1:k-1))} \tag{9.32}$$

If we define

$$\text{Odds}(x) = \frac{P(x)}{1 - P(x)}, \tag{9.33}$$

(9.32) turns into

$$\begin{aligned}&\text{Odds}(m \mid x(1:k), y(1:k))\\ &\quad = \frac{\text{Odds}(m \mid x(k), y(k))\ \text{Odds}(m \mid x(1:k-1), y(1:k-1))}{\text{Odds}(m)}.\end{aligned} \tag{9.34}$$

The corresponding log Odds representation of (9.34) is

$$\begin{aligned}&\log \text{Odds}(m \mid x(1:k), y(1:k))\\ &\quad = \log \text{Odds}(m \mid x(k), y(k)) - \log \text{Odds}(m)\\ &\qquad + \log \text{Odds}(m \mid x(1:k-1), y(1:k-1)).\end{aligned} \tag{9.35}$$

Please note that this equation also has a recursive structure similar to that of the recursive Bayesian filtering scheme described in Section 9.1. To incorporate a new scan into a given map we multiply its Odds ratio with the Odds ratio of a local map constructed from the most recent scan and divide it by the Odds ratio of the prior. Often it is assumed that the prior probability of m is 0.5. In this case the prior can be canceled so that (9.35) simplifies to

$$\begin{aligned}&\log \text{Odds}(m \mid x(1:k), y(1:k))\\ &\quad = \log \text{Odds}(m \mid x(k), y(k))\\ &\qquad + \log \text{Odds}(m \mid x(1:k-1), y(1:k-1)).\end{aligned} \tag{9.36}$$

To recover the occupancy probability from the Odds representation given in (9.34), we use the following law which can easily be derived from (9.33).

$$P(x) = \frac{\text{Odds}(x)}{1 + \text{Odds}(x)} \tag{9.37}$$

$$= \left[1 + \frac{1}{\text{Odds}(x)}\right]^{-1} \tag{9.38}$$

This leads to

$$\begin{aligned}&P(m \mid x(1:k), y(1:k))\\ &\quad = \left[1 + \frac{\text{Odds}(m)}{\text{Odds}(m \mid x(k), y(k))\ \text{Odds}(m \mid x(1:k-1), y(1:k-1))}\right]^{-1}\\ &\quad = \left[1 + \frac{1 - P(m \mid x(k), y(k))}{P(m \mid x(k), y(k))}\ \frac{P(m)}{1 - P(m)}\right.\\ &\qquad \left.\frac{1 - P(m \mid x(1:k-1), y(1:k-1))}{P(m \mid x(1:k-1), y(1:k-1))}\right]^{-1}.\end{aligned} \tag{9.39}$$

Algorithm 19 Occupancy grid mapping with known locations

Input: Sequence of measurements $y(1:k)$ and corresponding positions $x(1:k)$ and an initial belief $P_0(m)$ that the cells in the map are occupied

Output: Posterior $P_m = P(m \mid x(1:k), y(1:k))$ that the cells in the map are occupied

1: $P_m \leftarrow P_0(m)$
2: **for** $i \leftarrow 1$ to k **do**
3: $\quad P_m \leftarrow \left[1 + \frac{1-P(m \mid x(i),y(i))}{P(m \mid x(i),y(i))} \frac{P(m)}{1-P(m)} \frac{1-P_m}{P_m}\right]^{-1}$
4: **end for**

Algorithm 19 uses the recursive nature of (9.39) to compute the posterior $P(m \mid x(1:k), y(1:k))$. It receives as input the sequence of measurements $y(1:k)$ and the corresponding locations of the robot $x(1:k)$, as well as the initial probability $P_0(m)$ about the occupancy probability of the cells in the map. Typically, $P_0(m)$ will be initialized with the prior probability $P(m)$. If one wants to apply Algorithm 19 to multiple sequences of measurements, $P_0(m)$ can also be initialized with the output obtained from the previous application of the algorithm.

It remains to describe how we actually compute $P(m \mid x(k), y(k))$. Several techniques for determining this quantity have been presented. Whereas Moravec and Elfes [144, 325] used a probabilistic model to compute this quantity for ultrasound measurements, Thrun [411] applied a neural network to learn the appropriate interpretation of the measurements obtained with sonar sensors. The map depicted in the right image of figure 5.1 in chapter 5 has been computed with Thrun's approach. In this chapter we present a model that can be regarded as an approximate version of the approach described by Elfes [144].

One key assumption of occupancy probability grid-mapping techniques is that the individual cells of the map m can be considered independently. Accordingly, the posterior probability of m is computed as

$$P(m) = \prod_l P(m_l). \tag{9.40}$$

The advantage of this approach is that it suffices to describe how to update a single cell upon sensory input. Given this assumption, all we need to specify is the quantity $P(m_l \mid x(k), y(k))$ which is the probability that cell m_l is occupied given the measurement $y(k)$ and the state $x(k)$ of the robot.

The model $P(m_l \mid x(k), y(k))$ described here considers for each cell m_l the difference between the measured distance $y(k)$ and distance of m_l from $x(k)$. In the case of

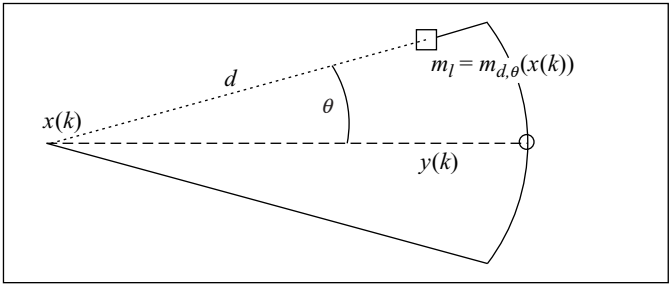

Figure 9.12 The occupancy probability of a cell $m_l = m_{d,\theta}(x(k))$ depends on the distance d to $x(k)$ and the angle θ to the optical axis of the cone.

ultrasound sensors the signal is typically emitted in a cone. To compute the occupancy probability of a cell m_l we therefore also consider the angle θ between the optical axis of the sensor and the ray going through m_l and $x(k)$ (see figure 9.12). The occupancy probability $P(m_l \mid x(k), y(k)) = P(m_{d,\theta}(x(k)) \mid y(k), x(k))$ of m_l is then computed using the following function, which can be regarded as an approximation of the mixture of Gaussians and linear functions applied by Elfes [144].

$$
\begin{aligned}
&P(m_{d,\theta}(x(k)) \mid y(k), x(k)) = P(m_{d,\theta}(x(k))) \\
&\quad + \begin{cases}
-s(y(k),\theta) & d < y(k) - d_1 \\
-s(y(k),\theta) + \frac{s(y(k),\theta)}{d_1}\,(d - y(k) + d_1) & d < y(k) + d_1 \\
s(y(k),\theta) & d < y(k) + d_2 \\
s(y(k),\theta) - \frac{s(y(k),\theta)}{d_3 - d_2}\,(d - y(k) - d_2) & d < y(k) + d_3 \\
0 & \text{otherwise.}
\end{cases}
\end{aligned} \tag{9.41}
$$

In this definition $s(y(k), \theta)$ is a function that computes the deviation of the occupancy probability from the prior occupancy probability $P(m)$ given the measured distance $y(k)$ and the angle θ between the cell, the sensor, and its optical axis. A common choice for $s(y(k), \theta)$ is a product of a linear function $g(y(k))$ and a Gaussian $\mathcal{N}(0, \sigma_\theta)$:

$$
s(y(k), \theta) = g(y(k))\, \mathcal{N}(0, \sigma_\theta) \tag{9.42}
$$

figure 9.13 plots these two components as they are used in the examples shown in this section. The variance σ_θ of the Gaussian is 0.05. Figure 9.14 plots $s(y(k), \theta)$ for $y(k) \in [0; 3m]$ and $\theta \in [-\frac{\pi}{24}; \frac{\pi}{24}]$. This angular range is identical to the opening angle of 15 degrees of the ultrasound sensors used to acquire the data of the examples presented here.

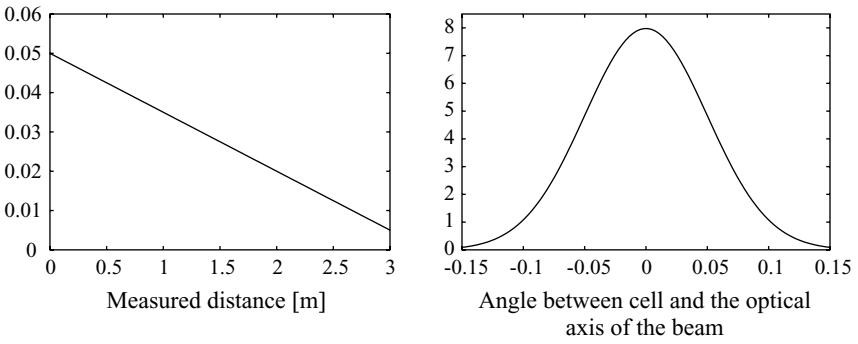

Figure 9.13 Functions used to compute the function $s(y(k), \theta)$: linear function (left) and Gaussian (right).

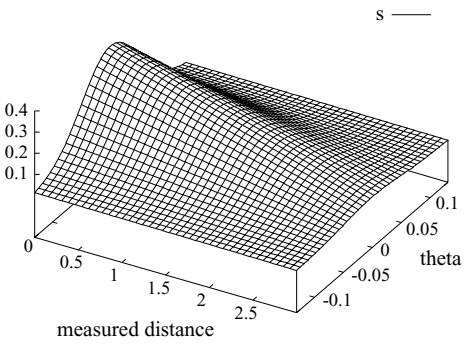

Figure 9.14 Function $s(y(k), \theta)$ used to model the deviation from the prior for cells in the sensor cone.

The constants d_1, d_2, and d_3 in (9.41) specify the interval in which the different linear functions of the piecewise linear approximation are valid (see also figure 9.15). The occupancy probability of cells lying between $x(k)$ and the arc from which the signal was reflected must be smaller than the prior probability for occupancy. In our model the occupancy probability of cells with $d < y(k) - d_1$ therefore is computed as $P(m_l) - s(y(k), \theta)$. The occupancy probability of cells whose distance to $x(k)$ is close to $y(k)$, i.e., for which $y(k) - d_1 \leq d < y(k) + d_1$, is computed by a linear function that increases with d. If a beam ends in a cell it is commonly assumed that the world is also occupied at least for a certain range behind that cell. In our model the occupancy probability therefore stays at a high but constant level $P(m_l) + s(y(k), \theta)$ for all cells whose distance lies between $y(k) + d_1$ and $y(k) + d_2$. Accordingly, the constants

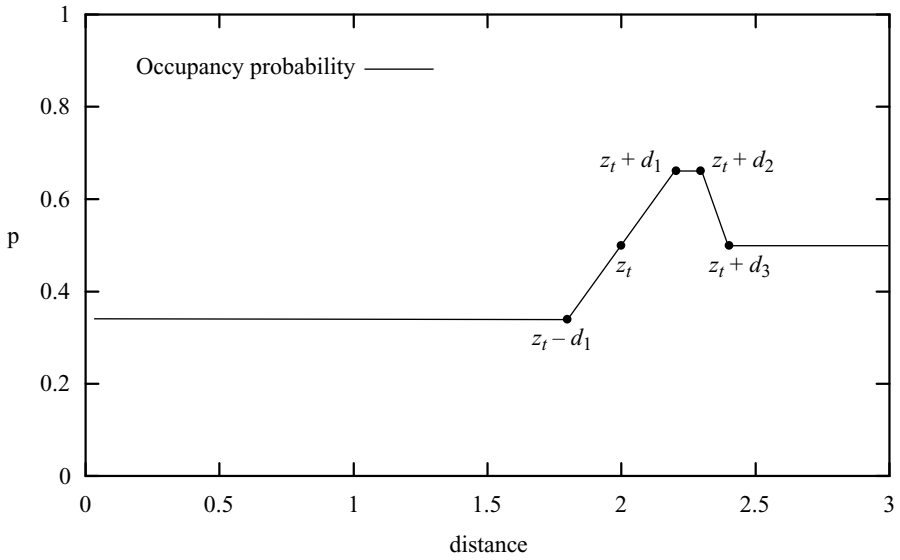

Figure 9.15 Probability $P(m_{d,\theta}(x(k))$ that a cell on the optical axis of the sensor ($\theta = 0$) is occupied depending on the distance of that cell from the sensor. The measured distance is $2m$.

d_1 and d_2 encode the average depth of obstacles. For distances d with $y(k) + d_2 \leq d < y(k) + d_3$, we assume that the occupancy probability linearly decreases to the prior occupancy probability $P(m_l)$. Finally, for cells $m_l = m_{d,\theta}(x(k))$ whose distance from $x(k)$ exceeds $y(k) + d_3$ we can safely assume that $P(m_{d,\theta}(x(k)) \mid y(k), x(k))$ equals the prior probability $P(m_l)$, since $y(k)$ does not give us any information about such cells. Figure 9.15 plots $P(m_{d,\theta}(x(k)) \mid y(k), x(k))$ for d ranging from 0 m to 3 m given that $y(k)$ is 2 m and that θ is 0 degrees. In this case, the value of $s(y(k), \theta)$ is approximately 0.16. We additionally assume that the prior probability $P(m_l) = P(m_{d,\theta}(x(k))) = 0.5$ for all d, θ, and $x(k)$.

Figure 9.16 shows three-dimensional plots of the resulting occupancy probabilities for measurements of 2.0 m and 2.5 m. In both plots the optical axis of the sensor cone is identical to the x-axis and the sensor is located in the origin of the coordinate frame. As can be seen from the figure, the occupancy probability is high for cells whose distance to $x(k)$ is close to $y(k)$. It decreases for cells with distance $y(k)$ from $x(k)$ and with increasing values of θ. Furthermore, it stays constant for cells that are immediately behind a cell that might have reflected the beam and linearly decreases to the prior probability afterward. For cells that are covered by the beam but did not reflect it, the occupancy probability is decreased.

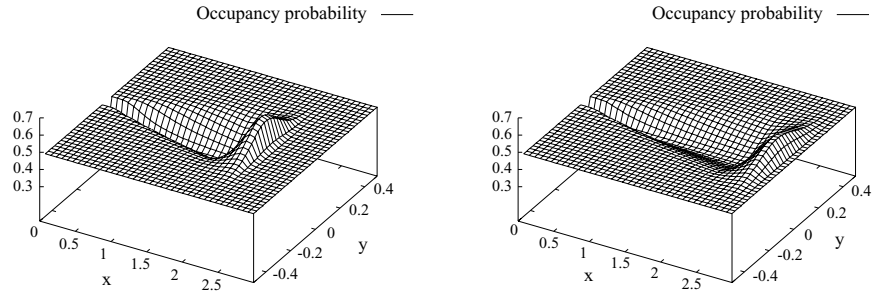

Figure 9.16 Local occupancy probability grids for a single ultrasound measurement of $y(k) = 2.0m$ (left) and $y(k) = 2.5m$ (right).

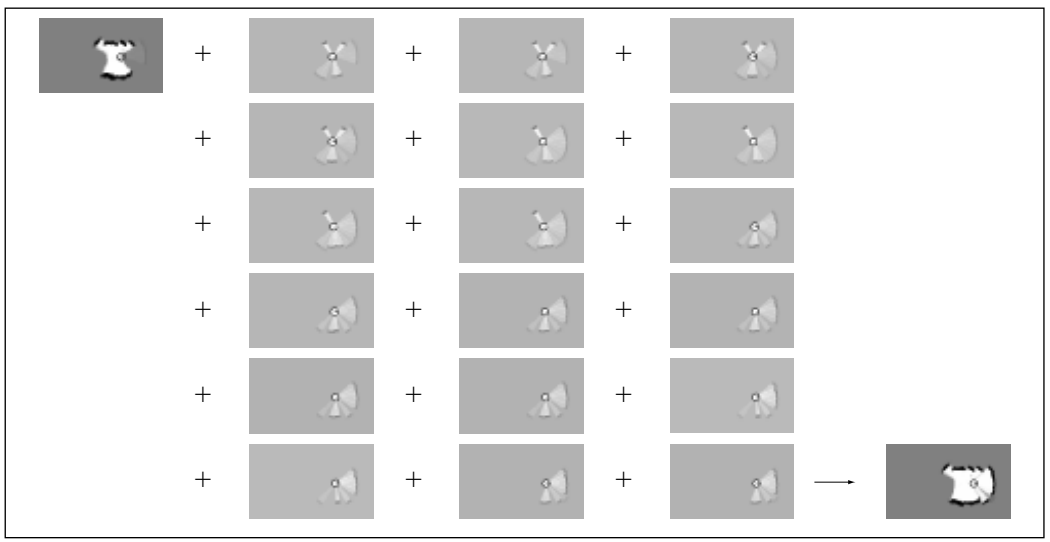

Figure 9.17 Incremental mapping in a corridor environment. The upper left image shows the initial map and the lower right image contains the resulting map. The maps in the three middle columns are the local maps built from the individual ultrasound scans perceived by the robot. Throughout this process, measurements beyond a 2.5 m radius have not been considered.

Figure 9.17 shows our sensor model in action for a sequence of measurements recorded with a B21r robot in a corridor environment. The upper-left corner shows a map that has been built from a sequence of ultrasound scans. Afterward the robot perceived a series of eighteen ultrasound scans each consisting of 24 measurements.

Figure 9.18 Occupancy probability map for the corridor of the Autonomous Intelligent Systems Lab at the University of Freiburg (left) and the corresponding maximum-likelihood map (right).

The occupancy probabilities for these eighteen scans are depicted in the three columns in the center of this figure. Note that during mapping we did not use measurements whose distance exceeded $2.5m$. The occupancy probability grid obtained by integrating the individual observations into the initial map is shown in the lower-right corner of this figure. As can be seen, the belief converges to an accurate representation of the corridor structure although the individual measurements show a high amount of uncertainty, as is usually the case for ultrasound sensors.

The left image of figure 9.18 shows the occupancy probabilities of the corridor environment obtained after incorporating all measurements of the data set used here. The map represents a 17 m long and 11 m wide part of a corridor environment including three rooms. The right image shows the corresponding maximum-likelihood map. This map is obtained from the occupancy probability grid by a simple clipping operation with a threshold of 0.5. The gray areas of the maximum-likelihood map correspond to cells that have not been sensed by the robot.

Let us briefly discuss some aspects that might be relevant to potential improvements of the models described here. The strongest restriction results from the assumption that all cells of the grid are considered independently. This independence assumption decomposes the high-dimensional state estimation problem into a set of onedimensional estimation problems. The independency of individual cells, however, is usually not justified in practice. For example, if the robot detects a door, then particular cells in the neighborhood need to be updated according to the specific shape of the door. Accordingly, techniques considering the individual cells of a grid independently might produce suboptimal solutions. One technique that addresses this problem has recently been presented by Thrun [413].

Additionally, occupancy probability grid maps assume that the environment has a binary structure, i.e., that every cell is either occupied or free. Occupancy probabilities cannot correctly represent situations in which a cell is only partly covered by an obstacle. Finally, most of the techniques, as well as our model, assume that the individual beams of the sensors can be considered independently when updating a map. This assumption also is not justified in practice, since neighboring beams of a scan often yield similar values. Accordingly, a robot ignoring this might become overly confident of the state of the environment.

9.2.2 Bayesian Simultaneous Localization and Mapping

In the previous subsection, we assumed that the robot always knows its position while it is mapping the environment. This assumption, however, is typically not justified, especially when a robot has to rely on its onboard sensors to determine its position due to the lack of a global positioning system, active beacons, or predefined landmarks. In such a situation, mapping turns into the so-called chicken and egg problem. Without a map the robot cannot determine its own position and without knowledge about its own position the robot cannot compute what its environment looks like. This is why this problem is often denoted as the SLAM problem (see also chapter 8).

In the past, research in the area of SLAM has led to two different types of approaches, each of which has its advantages and disadvantages [412]. The first class contains algorithms relying on the EKF to estimate joint posteriors over maps and robot locations [100, 128, 277]. These approaches provide a sound mathematical framework (see also chapter 8). However, they mainly have been applied in situations in which the environment contains predefined landmarks.

The second class of techniques considers the SLAM problem as a global optimization problem. For example, Lu and Milios [299] consider robot locations as random variables and derive constraints between locations from distances between overlapping range measurements and from odometry measurements. The constraints can be regarded as links in a network of springs, whose energy is to be minimized. Other approaches apply Dempster, Laird and Rubin's *expectation maximization,* or *EM* algorithm [127] to compute the maximum-likelihood estimate for the map and the locations of the robot. Examples of these kind of techniques can be found in [84, 126, 382, 417]. EM-based techniques have been applied successfully to mapping large cyclic environments with highly ambiguous features. However, they are inherently batch algorithms, requiring multiple passes through the entire data set. As a consequence, they usually cannot be applied when a robot has to map its environment online, i.e., while it is exploring it.

In probabilistic terms the problem of SLAM is to find the map and the robot positions which yield the best interpretation of the data gathered by the robot. As in

Section 9.1 the data consist of a stream of odometry measurements $u(0 : k-1)$ and perceptions of the environment $y(1 : k)$. According to Thrun [412], the mapping problem can be phrased as recursive Bayesian filtering for estimating the robot positions along with a map of the environment:

$$P(x(1:k), m \,|\, u(0:k-1), y(1:k)) = \alpha \; P(y(k) \,|\, x(k), m) \int \Big(P(x(k) \,|\, u(k-1), x(k-1)) \; P(x(1:k-1), m \,|\, u(0:k-2), y(1:k-1)) \Big) dx(1:k-1). \tag{9.43}$$

As in probabilistic localization (see Section 9.1) we assume that the odometry measurements are governed by a so-called probabilistic motion model $P(x(k) \,|\, x(k-1), u(k-1))$ which specifies the likelihood that the robot is at $x(k)$ given that it previously was at $x(k-1)$ and the motion $u(k-1)$ was measured. On the other hand, the observations follow the sensor model $P(y(k) \,|\, x(k), m)$, which defines for every possible location $x(k)$ in the environment the likelihood of the observation $y(k)$ given the map m.

Unfortunately, estimating the full posterior in (9.43) is not tractable in general. One approach is to apply incremental scan matching [180, 182, 365, 424]. The general idea of such approaches can be summarized as follows. At any point $k-1$ in time, the robot is given an estimate of its location $\hat{x}(k-1)$ and a map $\hat{m}(\hat{x}(1:k-1), y(1:k-1))$. After moving and taking a new measurement $y(k)$, the robot determines the most likely new location $\hat{x}(k)$ such that

$$\hat{x}(k) = \operatorname*{argmax}_{x(k)} \Big\{ P(y(k) \,|\, x(k), \hat{m}(\hat{x}(1:k-1), y(1:k-1))) \; P(x(k) \,|\, u(k-1), \hat{x}(k-1)) \Big\}. \tag{9.44}$$

It does this by trading off the consistency of the measurement with the map [first term on the right-hand side in (9.44)] and the consistency of the new location with the control action and the previous location [second term on the right-hand side in (9.44)]. The map is then extended by the new measurement $y(k)$, using the location $\hat{x}(k)$ as the location at which this measurement was taken. Popular techniques to determine $\hat{x}(k)$ in the context of laser range scans are the iterative-closest-point algorithm [46] or variants thereof.

The key limitation of scan-matching approaches lies in the greedy maximization step. Once the location $x(k)$ at time k has been computed it is not revised afterward so that the robot cannot recover from registration errors. Although scan matching techniques have been proven to be able to correct enormous errors in odometry, the resulting maps often are globally inconsistent. As an example consider figure 9.19

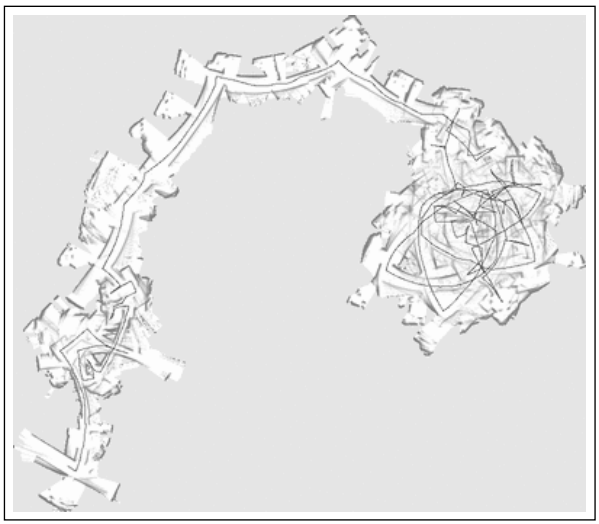

Figure 9.19 Map generated from raw odometry and laser range data gathered with a B21r robot.

which shows a map generated from raw odometry and laser range data obtained with a B21r robot. As can be seen from the figure, the robot suffers from serious errors in odometry so that the resulting map is useless without any correction. The size of this environment is 28 m $\times$ 28 m. When recording the data the robot traveled 491 m with an average speed of 0.19 m/s. Figure 9.20 shows the map created with the scan matching system presented by Hähnel, Schulz, and Burgard [182]. Although the local structures of the map appear to be very accurate, the map is globally inconsistent. For example, many structures like walls, doors, and such can be found several times and with a small offset between them.

To overcome this problem, alternative approaches have been developed. The key idea of these techniques is to maintain a posterior about the position of the vehicle. Whereas Gutmann and Konolige [178] used a discrete and grid-based approximation of the belief about the robots location, Thrun [412] applied a particle filter for this purpose. However, both approaches only maintain a single map and revise previous decisions whenever the robot closes a loop and returns to a previously visited place.

More recently, Murphy and coworkers [137,329] have proposed *Rao-Blackwellized particle filtering* as an efficient means to maintain multiple hypotheses during mapping. The key idea of this approach can be understood more easily when one considers the graphical model depicted in figure 9.21. If we know the map, the overall problem is transformed into a localization problem where the task is to estimate the location

Figure 9.20 Map obtained by applying a scan-matching approach to the same data used in figure 9.19.

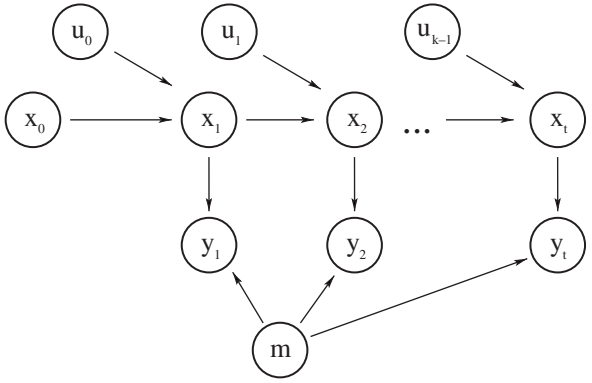

Figure 9.21 Graphical model of incremental probabilistic SLAM.

of the robot at each point in time. If, however, the locations are known, it remains solely to compute the map. Note that the knowledge of $x(1:k)$ is sufficient to figure out what the environment looks like, whereas $x(0)$ only determines the location of the map. Thus, if $x(1:k)$ is known but $x(0)$ is unknown, the robot can estimate its position relative to the map, but it cannot determine the location of the map.

The application of Rao-Blackwellized particle filtering to mapping is motivated by the observation that once the path $x(1:k)$ of the robot is known, the maximum-likelihood map can be computed analytically, e.g., using the method described in subsection 9.2.1. Therefore, the goal of SLAM with Rao-Blackwellized particle filters is to estimate the path of the robot using a particle filter and to analytically compute the map corresponding to that path. In practice this means that we use a set of particles to represent a posterior about potential paths of the robot. To each of these paths we associate an individual map that is computed based on the hypothesis that this path corresponds to the true path of the robot. The importance weight of a sample is proportional to the likelihood of the most recent observation given the map, which is computed based on the previous observations and the path of the robot according to that particular particle.

Note that Rao-Blackwellized particle filtering [136, 137] is a general technique to reduce the size of high-dimensional state estimation problems by marginalizing out parts of the state space. In this section we use this technique to develop an efficient solution to the SLAM problem.

Let us again consider the posterior $P(x(1:k), m \mid u(0:k-1), y(1:k))$ we want to estimate. If we apply the chain rule of probability theory, we obtain

$$\begin{aligned} &P(x(1:k), m \mid u(0:k-1), y(1:k)) \\ &\quad = P(m \mid x(1:k), y(1:k), u(0:k-1)) \\ &\qquad P(x(1:k) \mid y(1:k), u(0:k-1)). \end{aligned} \tag{9.45}$$

Obviously, we can safely assume that m is independent of $u(0:k-1)$ once we know the locations $x(1:k)$ of the robot, i.e.,

$$P(m \mid x(1:k), y(1:k), u(0:k-1)) = P(m \mid x(1:k), y(1:k)). \tag{9.46}$$

This leads to

$$\begin{aligned} &P(x(1:k), m \mid u(0:k-1), y(1:k)) \\ &\quad = P(m \mid x(1:k), y(1:k))\ P(x(1:k) \mid y(1:k), u(0:k-1)). \end{aligned} \tag{9.47}$$

In the previous section we saw that we can efficiently compute the posterior $P(m \mid x(1:k), y(1:k))$ for m given we know $x(1:k)$ and $y(1:k)$. Thus, all we need to do is to sample $P(x(1:k) \mid y(1:k), u(0:k-1))$ using a particle filter and compute for each particle the map that is associated to it.

We proceed as follows. Suppose $\mathcal{M}$ is a set of particles that represents the posterior about potential paths of the robot. In the beginning we assume that each particle starts at $[0, 0, 0]^T$, i.e., the robot is located at the origin of the coordinate system and its heading is 0. Let us furthermore denote the path associated with the jth particle by $h^{(j)}(1:k)$. As described above, once the path of the robot is known, we can directly compute the most likely map for that particle:

$$m^{(j)}(1:k-1) = \underset{m}{\operatorname{argmax}}\, P(m \mid h^{(j)}(1:k), y(1:k-1)) \tag{9.48}$$

Whenever an odometry measurement $u(i-1)$ is obtained, we proceed in the same way as we do in probabilistic localization. For each sample we compute the next location $x = x'_j$ of its path by sampling from $P(x \mid x_j, u(i-1))$. Note that—as in probabilistic localization—we in principle had to sample from $P(x \mid x_j, u(i-1), m^{(j)}(1:k-1))$, i.e., we also had to consider the map $m^{(j)}(1:k-1)$ associated with each sample. In practice, however, the map is often ignored for reasons of efficiency since computing $P(x \mid x_j, u(i-1), m^{(j)}(1:k-1))$ typically involves a time-consuming ray-casting operation in $m^{(j)}(1:k-1)$. Once we have computed for the jth particle both the map $m^{(j)}(1:k-1)$ and the new location x'_j, we are ready to compute the likelihood of the observation $y(k)$ and to use the resulting quantity as an importance weight during the resampling step. As a sensor model we can, e.g., choose the model described in section 9.1. The overall approach is realized by algorithm 20. Note that, according to the recursive structure of the problem, this algorithm can easily be extended for multiple sequences of sensory input. To do so, one simply has to ensure that the initialization is carried out using the output obtained from the previous application of the algorithm.

Please note that two aspects of this algorithm need to be implemented carefully to obtain the desired efficiency and convergence. If the map $m^{(j)}(1:k-1)$ is computed from scratch in every round, the resulting algorithm will be quadratic in k. On the other hand, maintaining a complete map for each individual particle (which also needs to be updated in each round) is inefficient with respect to memory. Additionally, it involves a time-consuming operation if the map associated with a sample has to be copied once for each of its successors in the resampling step. A popular approach to overcome this problem is to use treelike structures such as those proposed by Montemerlo et al [321], as well as by Parr and Eliazar [346]. Since typically many of the particles have larger parts of their history in common, the maps associated with the particles can efficiently be represented using trees.

An alternative although approximative approach to compute the maps associated with the individual samples is based on the observation that, in order to compute $P(y(k) \mid x_j, m^{(j)}(1:k-1))$, we only need to determine the part of the map $m^{(j)}(1:k-1)$ that is covered by $y(k)$. If we furthermore use only a limited number of

Algorithm 20 Simultaneous localization and mapping using Rao-Blackwellized particle filtering

Input: Sequence of measurements $y(1:k)$ and movements $u(0:k-1)$ and set $\mathcal{M}$ of N samples (x_j, ω_j)

Output: Posterior $P(x(1:k), m \mid u(0:k-1), y(1:k))$ represented by $\mathcal{M}$ about the path of the robot at time and the map

1: **for** $j \leftarrow 1$ to N **do**
2: $x_j \leftarrow (0, 0, 0)$
3: **end for**
4: **for** $i \leftarrow 1$ to k **do**
5: **for** $j \leftarrow 1$ to N **do**
6: compute a new state x by sampling according to $P(x \mid u(i-1), x_j)$.
7: $x_j \leftarrow x$
8: **end for**
9: $\eta \leftarrow 0$
10: **for** $j \leftarrow 1$ to N **do**
11: $w_j = P(y(i) \mid x_j, m^{(j)}(1:i-1)))$
12: $\eta = \eta + w_j$
13: **end for**
14: **for** $j \leftarrow 1$ to N **do**
15: $w_j = \eta^{-1} \cdot w_j$
16: **end for**
17: $\mathcal{M} = resample(\mathcal{M})$
18: **end for**

measurements from the history $h^{(j)}(1:k)$ we obtain an approximation of $m^{(j)}(1:k-1)$ that, if, additionally, spatial indices are used to compute the set of relevant scans from $h^{(j)}(1:k)$, can be computed in constant time. Thus, the overall complexity is constant for each particle. This approach has been successfully applied by Hähnel, Burgard, Fox, and Thrun [181] and has been used for the examples presented here.

A further aspect which turns out to be crucial to the success of the overall approach is the limitation of the number of particles that are needed. Since each particle possesses an individual map, the memory required by using Rao-Blackwellized filtering can be quite high, especially if many samples are needed to appropriately represent the posterior. One technique to reduce the number of necessary samples has recently been developed by Hähnel et al. [181]. In their approach consecutive laser range scans are converted into highly accurate odometry measurements. This way the uncertainty in the location of the robot is reduced so that fewer samples are needed to represent the posterior.

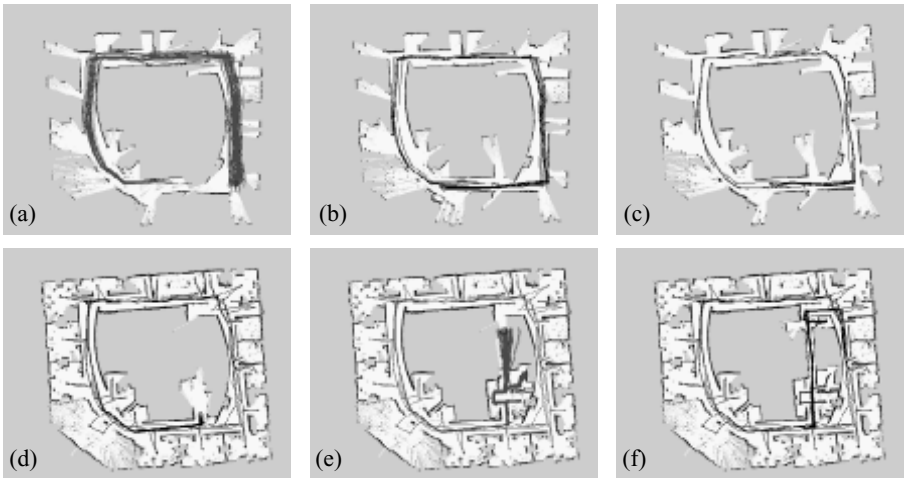

Figure 9.22 Sequence of maps corresponding to the particle with the highest accumulated importance weight during Rao-Blackwellized particle filtering of a large cyclic environment.

Figure 9.22 shows a Rao-Blackwellized particle filter for simultaneous localization and mapping in action. The individual figures illustrate the posterior about the robot's location as well as the map associated with the particle with the maximum accumulated importance factor. Image (a) shows the belief of the robot just before the robot is closing a loop. The second image (b) depicts the belief some steps later after the robot has closed the loop. As can be seen, the belief is more peaked due to the fact that particles whose observations do not match to their maps quickly die out when a loop is closed. Picture (c) shows a situation when the robot has moved around the loop for a second time. Please note that all figures also show the paths of all particles. A low number of different paths indicates that at the corresponding point in time, already many particles have a common history. In the situation depicted in image (d) the robot has visited all rooms in the building and enters a new corridor which imposes the task of closing another loop. The belief shortly before the robot closes this second loop is depicted in image (e). Image (f) shows the map and the particle histories after the robot finished its task. The resulting map is illustrated in figure 9.23.

After they have been demonstrated to be an efficient means for SLAM [137, 329], Rao-Blackwellized particle filters have been used with great success to learn large-scale maps of different types of environments. For example, Montemerlo et al. [321] have applied this technique to landmark-based mapping in which the locations of the individual landmarks are represented by Gaussians. In a more recent work [320] Montemerlo and Thrun extended this work to landmark-based mapping with uncertain

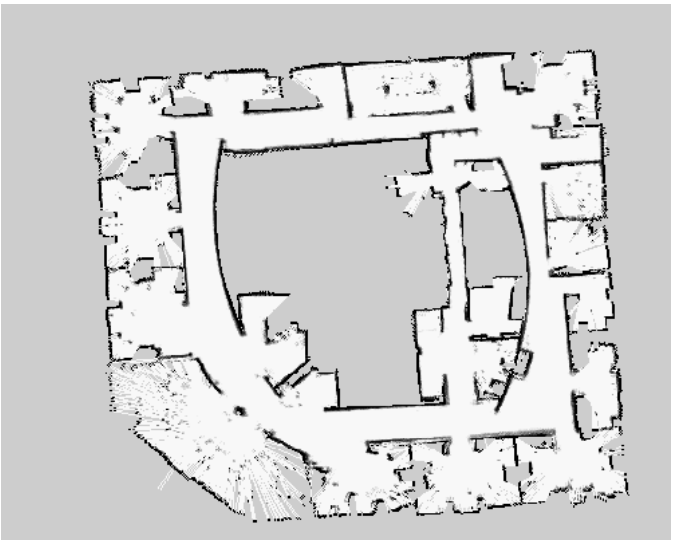

Figure 9.23 Resulting map obtained with Rao-Blackwellized particle filtering for SLAM.

data association. Additionally, this technique has been applied successfully to the simultaneous estimation of states of dynamic objects and the robot's locations [32, 322]. As mentioned above, new results present optimizations of this technique that allow the efficient application of Rao-Blackwellized particle filtering to SLAM with raw laser range scans [181, 346].

Problems

1. Prove the following variant of Bayes rule with background knowledge E:

   $$P(A \mid B, E) = \frac{P(B \mid A, E)P(A \mid E)}{P(B \mid E)}$$

2. Use (9.2) to rederive the equations for the Kalman filter.

3. Implement the sensor model for mobile robot localization using data from proximity sensors described in Subsection 9.1.5. Proceed in the following steps.

   (a) Generate a model of your robot's environment and implement a function that takes as input the location and sensing direction of a sensor and that generates as output the expected distance to the next obstacle in the direction of the measurement.

(b) Use (9.23) to generate for each of a discrete set of expected distances a histogram whose values represent the likelihood of the corresponding measured distance given that expected distance. Choose a discretization of 3 inches. To ensure that the values $h_{i,j}$ of each histogram sum up to one over all j for each i, choose appropriate values for δ_i.

4. Implement a motion model for sample-based mobile robot localization according to (9.19), (9.20), and (9.21). Realize a procedure that continuously reads odometry measurements from your robot and propagates a set of samples according to your motion model based on this input. Your program should generate sample sets similar to those shown in figure 9.6.

5. Implement probabilistic localization based on a particle filter using Algorithms 17 and 18. Combine the procedures developed in the previous two assignments and apply your algorithm to data obtained from your robot's odometry and from its proximity sensors.

6. Consider a robot that resides in a circular world consisting of ten different places that are numbered counterclockwise. The robot is unable to sense the number of its present place directly. However, places 0, 3, and 6 contain a distinct landmark, whereas all other places do not. All three of these landmarks look alike. The likelihood that the robot observes the landmark given it is in one of these places is 0.8. For all other places, the likelihood of observing the landmark is 0.4. For each place on the circle compute the probability that the robot is in that place given that the following sequence of actions is carried out deterministically and the following sequence of observations is obtained: The robot detects a landmark, moves 3 grid cells counterclockwise and detects a landmark, and then moves 4 grid cells counterclockwise and finally perceives no landmark.

7. Implement a program that simulates the world described in Question 6. Assume that the actions are nondeterministic: When moving from one place to a neighboring place, the robot succeeds with probability 0.8 but stays in the same place with probability 0.2. Run your algorithm to calculate the posterior probability over places (a) under the deterministic action model and (b) the non-deterministic action model.

8. Argue why, once the location $x(k)$ of the robot and the map m is known, the measurement $y(k)$ is independent of all previous measurements $y(1:k-1)$. Show why this independence does *not* hold if the map m is not given.

9. What happens if you apply the particle filter algorithm to a robot whose sensor is almost perfect? For example, what happens when the robot uses (almost) noise-free range sensors? Hint: For near-perfect sensors, the likelihood-function $P(y \mid x)$ will be extremely peaked, i.e., it will be almost zero for all measurements that are slightly off the correct noise-free value. How does the accuracy of the sensor affect the number of particles needed?

10. Implement a motion model for the particle filter algorithm which takes into account that the robot cannot move through obstacles. Run a simulation in which the robot starts in the center of an empty and quadratic room without doors. Suppose the robot moves forward d meters, then turns left and again moves d meters, where d is the width of the room.

11. Implement the circular world described in Question 6 using a particle filter algorithm.

12. Consider a robot that has to use its camera for localization. Discuss what models could be used to compute the likelihood $P(y \mid x)$ for vision-based robot localization.

13. Suppose the robot has only two kinds of observations y_1 and y_2. Furthermore, suppose that the occupancy probability of a particular cell covered by an observation of type y_1 is 0.8 and for observations of type y_2 is $1/3$. To which value will the occupancy probability of m_i converge if only one out of three measurements is y_1 and all others are y_2? *Hint:* Use (9.35) and calculate to which value the occupancy probability of m_i will converge if the number of measurements goes to infinity.

14. Discuss how the sensor model for $P(m_l \mid x(k), y(k))$ for occupancy grids changes if more accurate sensors are used and if the opening angle of the sonar cone becomes smaller.

15. Consider an environment which has the shape of a snailshell as depicted below. What will happen if you apply the Rao-Blackwellized particle filtering algorithm in such an environment? Consider robots with accurate/weak odometry and with accurate/noisy sensors.