2.4 BAYES FILTERS

2.4.1 The Bayes Filter Algorithm

The most general algorithm for calculating beliefs is given by the *Bayes filter* algorithm. This algorithm calculates the belief distribution bel from measurement and control data. We will first state the basic algorithm and elucidate it with a numerical example. After that, we will derive it mathematically from the assumptions made so far.

Table 2.1 depicts the basic Bayes filter in pseudo-algorithmic form. The Bayes filter is recursive, that is, the belief $bel(x_t)$ at time t is calculated from the belief $bel(x_{t-1})$ at time $t-1$. Its input is the belief bel at time $t-1$, along with the most recent control u_t and the most recent measurement z_t. Its output is the belief $bel(x_t)$ at time t. Table 2.1 only depicts a single step of the Bayes Filter algorithm: the *update rule*. This update rule is applied recursively, to calculate the belief $bel(x_t)$ from the belief $bel(x_{t-1})$, calculated previously.

The Bayes filter algorithm possesses two essential steps. In Line 3, it processes the control u_t. It does so by calculating a belief over the state x_t based on the prior belief over state x_{t-1} and the control u_t. In particular, the belief $\overline{bel}(x_t)$ that the robot assigns to state x_t is obtained by the integral (sum) of the product of two distributions: the prior assigned to x_{t-1}, and the probability that control u_t induces a transition from x_{t-1} to x_t. The reader may recognize the similarity of this update step to Equation (2.12). As noted above, this update step is called the control update, or *prediction*.

The second step of the Bayes filter is called the measurement update. In Line 4, the Bayes filter algorithm multiplies the belief $\overline{bel}(x_t)$ by the probability that the measurement z_t may have been observed. It does so for each hypothetical posterior state x_t. As will become apparent further below when actually deriving the basic filter equations, the resulting product is generally not a probability, that is, it may not integrate to 1. Hence, the result is normalized, by virtue of the normalization constant η. This leads to the final belief $bel(x_t)$, which is returned in Line 6 of the algorithm.

To compute the posterior belief recursively, the algorithm requires an initial belief $bel(x_0)$ at time $t = 0$ as boundary condition. If one knows the value of x_0 with certainty, $bel(x_0)$ should be initialized with a point mass distribution that centers all probability mass on the correct value of x_0, and assigns zero probability anywhere else. If one is entirely ignorant about the initial value x_0, $bel(x_0)$ may be initialized using a uniform distribution over the domain of x_0 (or related distribution from the Dirichlet family of distributions). Partial knowledge of the initial value x_0 can be

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

Table 2.1 The general algorithm for Bayes filtering.

expressed by non-uniform distributions; however, the two cases of full knowledge and full ignorance are the most common ones in practice.

The algorithm Bayes filter can only be implemented in the form stated here for very simple estimation problems. In particular, we either need to be able to carry out the integration in Line 3 and the multiplication in Line 4 in closed form, or we need to restrict ourselves to finite state spaces, so that the integral in Line 3 becomes a (finite) sum.

2.4.2 Example

Our illustration of the Bayes filter algorithm is based on the scenario in Figure 2.2, which shows a robot estimating the state of a door using its camera. To make this problem simple, let us assume that the door can be in one of two possible states, open or closed, and that only the robot can change the state of the door. Let us furthermore assume that the robot does not know the state of the door initially. Instead, it assigns equal prior probability to the two possible door states:

$$bel(X_0 = \mathbf{open}) = 0.5 \tag{2.37}$$
$$bel(X_0 = \mathbf{closed}) = 0.5 \tag{2.38}$$

Let us furthermore assume the robot's sensors are noisy. The noise is characterized by the following conditional probabilities:

$$\begin{aligned} p(Z_t = \mathbf{sense\_open} \mid X_t = \mathbf{is\_open}) &= 0.6 \\ p(Z_t = \mathbf{sense\_closed} \mid X_t = \mathbf{is\_open}) &= 0.4 \end{aligned} \tag{2.39}$$

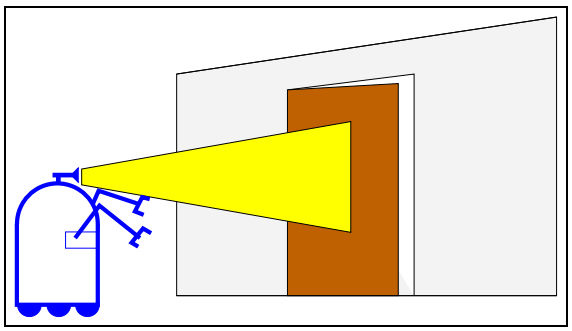

Figure 2.2 A mobile robot estimating the state of a door.

and

$$\begin{aligned} p(Z_t = \textbf{sense\_open} \mid X_t = \textbf{is\_closed}) &= 0.2 \\ p(Z_t = \textbf{sense\_closed} \mid X_t = \textbf{is\_closed}) &= 0.8 \end{aligned} \tag{2.40}$$

These probabilities suggest that the robot's sensors are relatively reliable in detecting a *closed* door, in that the error probability is 0.2. However, when the door is open, it has a 0.4 probability of a false measurement.

Finally, let us assume the robot uses its manipulator to push the door open. If the door is already open, it will remain open. It it is closed, the robot has a 0.8 chance that it will be open afterwards:

$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_open}) = 1$$

$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_open}) = 0 \tag{2.41}$$

$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_closed}) = 0.8$$

$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_closed}) = 0.2 \tag{2.42}$$

It can also choose not to use its manipulator, in which case the state of the world does not change. This is stated by the following conditional probabilities:

$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_open}) = 1$$

$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_open}) = 0 \tag{2.43}$$

$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_closed}) = 0$$

$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_closed}) = 1 \tag{2.44}$$

Suppose at time t, the robot takes no control action but it senses an open door. The resulting posterior belief is calculated by the Bayes filter using the prior belief $bel(X_0)$, the control $u_1 = \textbf{do\_nothing}$, and the measurement $\textbf{sense\_open}$ as input. Since the state space is finite, the integral in Line 3 turns into a finite sum:

$$\begin{aligned}
\overline{bel}(x_1) &= \int p(x_1 \mid u_1, x_0)\, bel(x_0)\, dx_0 \\
&= \sum_{x_0} p(x_1 \mid u_1, x_0)\, bel(x_0) \\
&= p(x_1 \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_open})\, bel(X_0 = \textbf{is_open}) \\
&\quad + p(x_1 \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_closed})\, bel(X_0 = \textbf{is_closed})
\end{aligned} \tag{2.45}$$

We can now substitute the two possible values for the state variable X_1. For the hypothesis $X_1 = \textbf{is\_open}$, we obtain

$$\begin{aligned}
&\overline{bel}(X_1 = \textbf{is_open}) \\
&= p(X_1 = \textbf{is_open} \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_open})\, bel(X_0 = \textbf{is_open}) \\
&\quad + p(X_1 = \textbf{is_open} \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_closed})\, bel(X_0 = \textbf{is_closed}) \\
&= 1 \cdot 0.5 + 0 \cdot 0.5 \;=\; 0.5
\end{aligned} \tag{2.46}$$

Likewise, for $X_1 = \textbf{is\_closed}$ we get

$$\begin{aligned}
&\overline{bel}(X_1 = \textbf{is_closed}) \\
&= p(X_1 = \textbf{is_closed} \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_open})\, bel(X_0 = \textbf{is_open}) \\
&\quad + p(X_1 = \textbf{is_closed} \mid U_1 = \textbf{do_nothing}, X_0 = \textbf{is_closed})\, bel(X_0 = \textbf{is_closed}) \\
&= 0 \cdot 0.5 + 1 \cdot 0.5 \;=\; 0.5
\end{aligned} \tag{2.47}$$

The fact that the belief $\overline{bel}(x_1)$ equals our prior belief $bel(x_0)$ should not surprise, as the action $\textbf{do\_nothing}$ does not affect the state of the world; neither does the world change over time by itself in our example.

Incorporating the measurement, however, changes the belief. Line 4 of the Bayes filter algorithm implies

$$bel(x_1) = \eta\, p(Z_1 = \textbf{sense\_open} \mid x_1)\, \overline{bel}(x_1)\,. \tag{2.48}$$

For the two possible cases, $X_1 = \textbf{is\_open}$ and $X_1 = \textbf{is\_closed}$, we get

$$\begin{aligned} & bel(X_1 = \textbf{is\_open}) \\ &= \eta\, p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_open})\, \overline{bel}(X_1 = \textbf{is\_open}) \\ &= \eta\, 0.6 \cdot 0.5 \;=\; \eta\, 0.3 \end{aligned} \tag{2.49}$$

and

$$\begin{aligned} & bel(X_1 = \textbf{is\_closed}) \\ &= \eta\, p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_closed})\, \overline{bel}(X_1 = \textbf{is\_closed}) \\ &= \eta\, 0.2 \cdot 0.5 \;=\; \eta\, 0.1 \end{aligned} \tag{2.50}$$

The normalizer η is now easily calculated:

$$\eta \;=\; (0.3 + 0.1)^{-1} \;=\; 2.5 \tag{2.51}$$

Hence, we have

$$\begin{aligned} bel(X_1 = \textbf{is\_open}) &= 0.75 \\ bel(X_1 = \textbf{is\_closed}) &= 0.25 \end{aligned} \tag{2.52}$$

This calculation is now easily iterated for the next time step. As the reader easily verifies, for $u_2 = \textbf{push}$ and $z_2 = \textbf{sense\_open}$ we get

$$\begin{aligned} \overline{bel}(X_2 = \textbf{is\_open}) &= 1 \cdot 0.75 + 0.8 \cdot 0.25 \;=\; 0.95 \\ \overline{bel}(X_2 = \textbf{is\_closed}) &= 0 \cdot 0.75 + 0.2 \cdot 0.25 \;=\; 0.05\,, \end{aligned} \tag{2.53}$$

and

$$\begin{aligned} bel(X_2 = \textbf{is\_open}) &= \eta\, 0.6 \cdot 0.95 \;\approx\; 0.983 \\ bel(X_2 = \textbf{is\_closed}) &= \eta\, 0.2 \cdot 0.05 \;\approx\; 0.017\,. \end{aligned} \tag{2.54}$$

At this point, the robot believes that with 0.983 probability the door is open, hence both its measurements were correct. At first glance, this probability may appear to be

sufficiently high to simply accept this hypothesis as the world state and act accordingly. However, such an approach may result in unnecessarily high costs. If mistaking a closed door for an open one incurs costs (e.g., the robot crashes into a door), considering both hypotheses in the decision making process will be essential, as unlikely as one of them may be. Just imagine flying an aircraft on auto pilot with a perceived chance of 0.983 for not crashing!

2.4.3 Mathematical Derivation of the Bayes Filter

The correctness of the Bayes filter algorithm is shown by induction. To do so, we need to show that it correctly calculates the posterior distribution $p(x_t \mid z_{1:t}, u_{1:t})$ from the corresponding posterior one time step earlier, $p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})$. The correctness follows then by induction under the assumption that we correctly initialized the prior belief $bel(x_0)$ at time $t = 0$.

Our derivation requires that the state x_t is complete, as defined in Section 2.3.1, and it requires that controls are chosen at random. The first step of our derivation involves the application of Bayes rule (2.23) to the target posterior:

$$\begin{aligned} p(x_t \mid z_{1:t}, u_{1:t}) &= \frac{p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\, p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} \\ &= \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t})\, p(x_t \mid z_{1:t-1}, u_{1:t}) \end{aligned} \tag{2.55}$$

We now exploit the assumption that our state is complete. In Section 2.3.1, we defined a state x_t to be complete if no variables prior to x_t may influence the stochastic evolution of future states. In particular, if we (hypothetically) knew the state x_t and were interested in predicting the measurement z_t, no past measurement or control would provide us additional information. In mathematical terms, this is expressed by the following conditional independence:

$$p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)\,. \tag{2.56}$$

Such a statement is another example of *conditional independence*. It allows us to simplify (2.55) as follows:

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{2.57}$$

and hence

$$bel(x_t) \quad = \quad \eta \, p(z_t \mid x_t) \, \overline{bel}(x_t) \tag{2.58}$$

This equation is implemented in Line 4 of the Bayes filter algorithm in Table 2.1.

Next, we expand the term $\overline{bel}(x_t)$, using (2.12):

$$\begin{aligned} \overline{bel}(x_t) \quad &= \quad p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \quad \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \, p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \end{aligned} \tag{2.59}$$

Once again, we exploit the assumption that our state is complete. This implies if we know x_{t-1}, past measurements and controls convey no information regarding the state x_t. This gives us

$$p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \quad = \quad p(x_t \mid x_{t-1}, u_t) \tag{2.60}$$

Here we retain the control variable u_t, since it does *not* predate the state x_{t-1}. Finally, we note that the control u_t can safely be omitted from the set of conditioning variables in $p(x_{t-1} \mid z_{1:t-1}, u_{1:t})$ for randomly chosen controls. This gives us the recursive update equation

$$\overline{bel}(x_t) \quad = \quad \int p(x_t \mid x_{t-1}, u_t) \, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \, dx_{t-1} \tag{2.61}$$

As the reader easily verifies, this equation is implemented by Line 3 of the Bayes filter algorithm in Table 2.1. To summarize, the Bayes filter algorithm calculates the posterior over the state x_t conditioned on the measurement and control data up to time t. The derivation assumes that the world is Markov, that is, the state is complete.

Any concrete implementation of this algorithm requires three probability distributions: The initial belief $p(x_0)$, the measurement probability $p(z_t \mid x_t)$, and the state transition probability $p(x_t \mid u_t, x_{t-1})$. We have not yet specified these densities, but will do so in later chapters (Chapters 5 and **??**). Additionally, we also need a representation for the belief $bel(x_t)$, which will also be discussed further below.

2.4.4 The Markov Assumption

A word is in order on the Markov assumption, or the complete state assumption, since it plays such a fundamental role in the material presented in this book. The Markov assumption postulates that past and future data are independent if one knows the current state x_t. To see how severe an assumption this is, let us consider our example of mobile robot localization. In mobile robot localization, x_t is the robot's pose, and Bayes filters are applied to estimate the pose relative to a fixed map. The following factors may have a systematic effect on sensor readings. Thus, they induce violations of the Markov assumption:

- Unmodeled dynamics in the environment not included in x_t (e.g., moving people and their effects on sensor measurements in our localization example),
- inaccuracies in the probabilistic models $p(z_t \mid x_t)$ and $p(x_t \mid u_t, x_{t-1})$,
- approximation errors when using approximate representations of belief functions (e.g., grids or Gaussians, which will be discussed below), and
- software variables in the robot control software that influence multiple control selection (e.g., the variable "target location" typically influences an entire sequence of control commands).

In principle, many of these variables can be included in state representations. However, incomplete state representations are often preferable to more complete ones to reduce the computational complexity of the Bayes filter algorithm. In practice Bayes filters have been found to be surprisingly robust to such violations. As a general rule of thumb, one has to exercise care when defining the state x_t, so that the effect of unmodeled state variables has close-to-random effects.

2.5 REPRESENTATION AND COMPUTATION

In probabilistic robotics, Bayes filters are implemented in several different ways. As we will see in the next two chapters, there exist quite a variety of techniques and algorithms that are all derived from the Bayes filter. Each such technique relies on different assumptions regarding the measurement and state transition probabilities and the initial belief. Those assumptions then give rise to different types of posterior distributions, and the algorithms for computing (or approximating) those have different

computational characteristics. As a general rule of thumb, exact techniques for calculating beliefs exist only for highly specialized cases; in general robotics problems, beliefs have to be approximated. The nature of the approximation has important ramifications on the complexity of the algorithm. Finding a suitable approximation is usually a challenging problem, with no unique best answer for all robotics problems.

When choosing an approximation, one has to trade off a range of properties:

1. **Computational efficiency.** Some approximations, such as linear Gaussian approximations that will be discussed further below, make it possible to calculate beliefs in time polynomial in the dimension of the state space. Others may require exponential time. Particle based techniques, discussed further below, have an *any-time* characteristic, enabling them to trade off accuracy with computational efficiency.

2. **Accuracy of the approximation.** Some approximations can approximate a wider range of distributions more tightly than others. For example, linear Gaussian approximations are limited to unimodal distributions, whereas histogram representations can approximate multi-modal distributions, albeit with limited accuracy. Particle representations can approximate a wide array of distributions, but the number of particles needed to attain a desired accuracy can be large.

3. **Ease of implementation.** The difficulty of implementing probabilistic algorithms depends on a variety of factors, such as the form of the measurement probability $p(z_t \mid x_t)$ and the state transition probability $p(x_t \mid u_t, x_{t-1})$. Particle representations often yield surprisingly simple implementations for complex nonlinear systems—one of the reasons for their recent popularity.

The next two chapters will introduce concrete implementable algorithms, which fare quite differently relative to the criteria described above.

2.6 SUMMARY

In this section, we introduced the basic idea of Bayes filters in robotics, as a means to estimate the state of an environment (which may include the state of the robot itself).

- The interaction of a robot and its environment is modeled as a coupled dynamical system, in which the robot can manipulate its environment by choosing controls, and in which it can perceive its environment through sensor measurements.

- In probabilistic robotics, the dynamics of the robot and its environment are characterized in the form of two probabilistic laws: the state transition distribution, and the measurement distribution. The state transition distribution characterizes how state changes over time, possible as the effect of a robot control. The measurement distribution characterizes how measurements are governed by states. Both laws are probabilistic, accounting for the inherent uncertainty in state evolution and sensing.

- The *belief* of a robot is the posterior distribution over the state of the environment (including the robot state), given all past sensor measurements and all past controls. The *Bayes filter* is the principal algorithm for calculating the belief in robotics. The Bayes filter is recursive; the belief at time t is calculated from the belief at time $t-1$.

- The Bayes filter makes a *Markov assumption* that specifies that the state is a complete summary of the past. This assumption implies the belief is sufficient to represent the past history of the robot. In robotics, the Markov assumption is usually only an approximation. We identified conditions under which it is violated.

- Since the Bayes filter is not a practical algorithm, in that it cannot be implemented on a digital computer, probabilistic algorithms use tractable approximations. Such approximations may be evaluated according to different criteria, relating to their accuracy, efficiency, and ease of implementation.

The next two chapters discuss two popular families of recursive state estimation techniques that are both derived from the Bayes filter.

2.7 BIBLIOGRAPHICAL REMARKS

3

GAUSSIAN FILTERS

3.1 INTRODUCTION

This chapter describes an important family of recursive state estimators, collectively called *Gaussian Filters*. Historically, Gaussian filters constitute the earliest tractable implementations of the Bayes filter for continuous spaces. They are also by far the most popular family of techniques to date—despite a number of shortcomings.

Gaussian techniques all share the basic idea that beliefs are represented by multivariate normal distributions. We already encountered a definition of the multivariate normal distribution in Equation (2.4), which is restated here:

$$p(x) \quad = \quad \det(2\pi\Sigma)^{-\frac{1}{2}} \; \exp\left\{-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{3.1}$$

This density over the variable x is characterized by two sets of parameters: The mean μ and the covariance Σ. The mean μ is a vector that possesses the same dimensionality as the state x. The covariance is a quadratic matrix that is symmetric and positive-semidefinite. Its dimension is the dimensionality of the state x squared. Thus, the number of elements in the covariance matrix depends quadratically on the number of elements in the state vector.

The commitment to represent the posterior by a Gaussian has important ramifications. Most importantly, Gaussians are unimodal, that is, they posses a single maximum. Such a posterior is characteristic of many tracking problems in robotics, in which the posterior is focused around the true state with a small margin of uncertainty. Gaussian posteriors are a poor match for many global estimation problems in which many distinct hypotheses exist, each of which forming its own mode in the posterior.

The representation of a Gaussian by its mean and covariance is called the *moments representation.* This is because the mean and covariance are the first and second moments of a probability distribution; all other moments are zero for normal distributions. In this chapter, we will also discuss an alternative representation, called *canonical representation*, or sometimes *natural representation*. Both representations, the moments and the canonical representations, are functionally equivalent in that a bijective mapping exists that transforms one into the other (and back). However, they lead to filter algorithms with orthogonal computational characteristics.

This chapter introduces the two basic Gaussian filter algorithms.

- Section 3.2 describes the Kalman filter, which implements the Bayes filter using the moments representation for a restricted class of problems with linear dynamics and measurement functions.
- The Kalman filter is extended to nonlinear problems in Section 3.3, which describes the extended Kalman filter.
- Section 3.4 describes the information filter, which is the dual of the Kalman filter using the canonical representation of Gaussians.

3.2 THE KALMAN FILTER

3.2.1 Linear Gaussian Systems

Probably the best studied technique for implementing Bayes filters is the *Kalman filter (KF)*. The Kalman filter was invented in the 1950s by Rudolph Emil Kalman, as a technique for filtering and prediction in linear systems. The Kalman filter implements belief computation for continuous states. It is not applicable to discrete or hybrid state spaces.

The Kalman filter represents beliefs by the moments representation: At time t, the belief is represented by the the mean μ_t and the covariance Σ_t. Posteriors are Gaussian if the following three properties hold, in addition to the Markov assumptions of the Bayes filter.

1. The next state probability $p(x_t \mid u_t, x_{t-1})$ must be a *linear* function in its arguments with added Gaussian noise. This is expressed by the following equation:

$$x_t \quad = \quad A_t x_{t-1} + B_t u_t + \varepsilon_t \; . \tag{3.2}$$

Here x_t and x_{t-1} are state vectors, and u_t is the control vector at time t. In our notation, both of these vectors are vertical vectors, that is, they are of the form

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{pmatrix} \quad \text{and} \quad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{pmatrix} . \tag{3.3}$$

A_t and B_t are matrices. A_t is a square matrix of size $n \times n$, where n is the dimension of the state vector x_t. B_t is of size $n \times m$, with m being the dimension of the control vector u_t. By multiplying the state and control vector with the matrices A_t and B_t, respectively, the state transition function becomes *linear* in its arguments. Thus, Kalman filters assume linear system dynamics.

The random variable ε_t in (3.2) is a Gaussian random vector that models the randomness in the state transition. It is of the same dimension as the state vector. Its mean is zero and its covariance will be denoted R_t. A state transition probability of the form (3.2) is called a *linear Gaussian*, to reflect the fact that it is linear in its arguments with additive Gaussian noise.

Equation (3.2) defines the state transition probability $p(x_t \mid u_t, x_{t-1})$. This probability is obtained by plugging Equation (3.2) into the definition of the multivariate normal distribution (3.1). The mean of the posterior state is given by $A_t x_{t-1} + B_t u_t$ and the covariance by R_t:

$$\begin{aligned} & p(x_t \mid u_t, x_{t-1}) \\ &= \det(2\pi R_t)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\} \end{aligned} \tag{3.4}$$

2. The measurement probability $p(z_t \mid x_t)$ must also be *linear* in its arguments, with added Gaussian noise:

$$z_t = C_t x_t + \delta_t . \tag{3.5}$$

Here C_t is a matrix of size $k \times n$, where k is the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t. The measurement probability is thus given by the following multivariate normal distribution:

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(z_t - C_t\, x_t)^T\, Q_t^{-1}\, (z_t - C_t\, x_t)\right\} \tag{3.6}$$

1: **Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):**
2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
3: $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

Table 3.1 The Kalman filter algorithm for linear Gaussian state transitions and measurements.

3. Finally, the initial belief $bel(x_0)$ must be normal distributed. We will denote the mean of this belief by μ_0 and the covariance by Σ_0:

$$bel(x_0) \quad = \quad p(x_0) \quad = \quad \det(2\pi\Sigma_0)^{-\frac{1}{2}}\ \exp\left\{-\tfrac{1}{2}\ (x_0 - \mu_0)^T \Sigma_0^{-1} (x_0 - \mu_0)\right\}$$

These three assumptions are sufficient to ensure that the posterior $bel(x_t)$ is always a Gaussian, for any point in time t. The proof of this non-trivial result can be found below, in the mathematical derivation of the Kalman filter (Section 3.2.4).

3.2.2 The Kalman Filter Algorithm

The Kalman filter algorithm is depicted in Table 3.1. Kalman filters represent the belief $bel(x_t)$ at time t by the mean μ_t and the covariance Σ_t. The input of the Kalman filter is the belief at time $t-1$, represented by μ_{t-1} and Σ_{t-1}. To update these parameters, Kalman filters require the control u_t and the measurement z_t. The output is the belief at time t, represented by μ_t and Σ_t.

In Lines 2 and 3, the predicted belief $\bar{\mu}$ and $\bar{\Sigma}$ is calculated representing the belief $\overline{bel}(x_t)$ one time step later, but before incorporating the measurement z_t. This belief is obtained by incorporating the control u_t. The mean is updated using the deterministic version of the state transition function (3.2), with the mean μ_{t-1} substituted for the state x_{t-1}. The update of the covariance considers the fact that states depend on previous states through the linear matrix A_t. This matrix is multiplied twice into the covariance, since the covariance is a quadratic matrix.

The belief $\overline{bel}(x_t)$ is subsequently transformed into the desired belief $bel(x_t)$ in Lines 4 through 6, by incorporating the measurement z_t. The variable K_t, computed in Line 4 is called *Kalman gain*. It specifies the degree to which the measurement is incorporated into the new state estimate. Line 5 manipulates the mean, by adjusting it in proportion to the Kalman gain K_t and the deviation of the actual measurement, z_t, and the measurement predicted according to the measurement probability (3.5). Finally, the new covariance of the posterior belief is calculated in Line 6, adjusting for the information gain resulting from the measurement.

The Kalman filter is computationally quite efficient. For today's best algorithms, the complexity of matrix inversion is approximately $O(d^{2.8})$ for a matrix of size $d \times d$. Each iteration of the Kalman filter algorithm, as stated here, is lower bounded by (approximately) $O(k^{2.8})$, where k is the dimension of the measurement vector z_t. This (approximate) cubic complexity stems from the matrix inversion in Line 4. It is also at least in $O(n^2)$, where n is the dimension of the state space, due to the multiplication in Line 6 (the matrix $K_t C_t$ may be sparse). In many applications—such as the robot mapping applications discussed in later chapters—-the measurement space is much lower dimensional than the state space, and the update is dominated by the $O(n^2)$ operations.

3.2.3 Illustration

Figure 3.2 illustrates the Kalman filter algorithm for a simplistic one-dimensional localization scenario. Suppose the robot moves along the horizontal axis in each diagram in Figure 3.2. Let the prior over the robot location be given by the normal distribution shown in Figure 3.2a. The robot queries its sensors on its location (e.g., a GPS system), and those return a measurement that is centered at the peak of the bold Gaussian in Figure 3.2b. This bold Gaussian illustrates this measurement: Its peak is the value predicted by the sensors, and its width (variance) corresponds to the uncertainty in the measurement. Combining the prior with the measurement, via Lines 4 through 6 of the Kalman filter algorithm in Table 3.1, yields the bold Gaussian in Figure 3.2c. This belief's mean lies between the two original means, and its uncertainty radius is smaller than both contributing Gaussians. The fact that the residual uncertainty is smaller than the contributing Gaussians may appear counter-intuitive, but it is a general characteristic of information integration in Kalman filters.

Next, assume the robot moves towards the right. Its uncertainty grows due to the fact that the next state transition is stochastic. Lines 2 and 3 of the Kalman filter provides us with the Gaussian shown in bold in Figure 3.2d. This Gaussian is shifted by the amount the robot moved, and it is also wider for the reasons just explained. Next, the

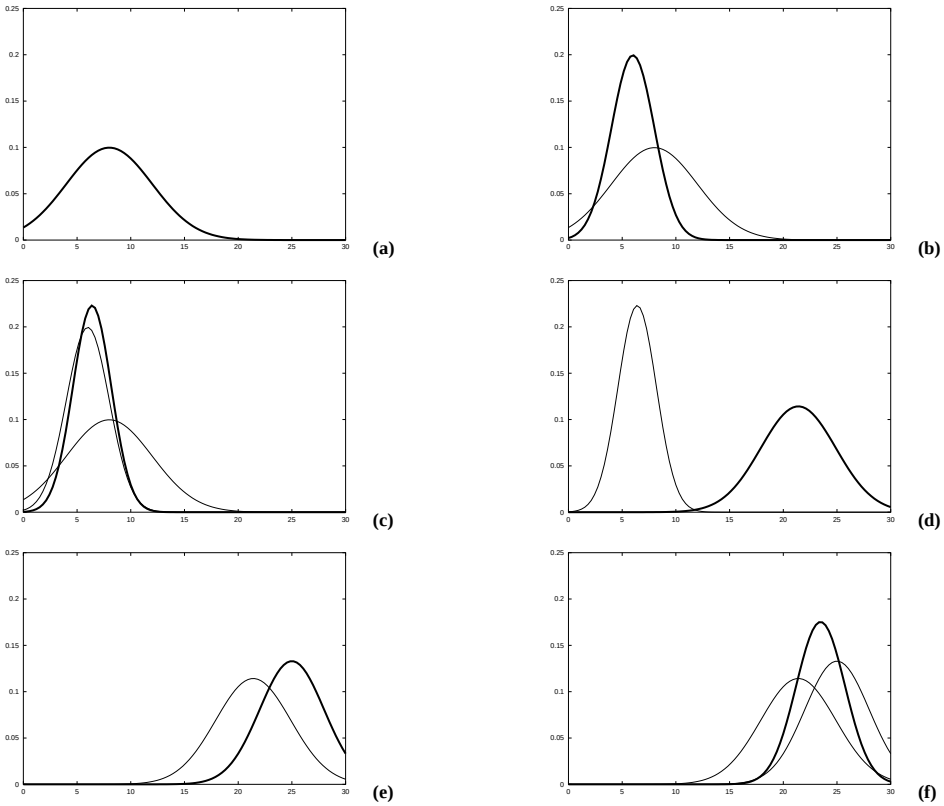

Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

robot receives a second measurement illustrated by the bold Gaussian in Figure 3.2e, which leads to the posterior shown in bold in Figure 3.2f.

As this example illustrates, the Kalman filter alternates a *measurement update step* (Lines 5-7), in which sensor data is integrated into the present belief, with a *prediction step* (or control update step), which modifies the belief in accordance to an action. The update step decreases and the prediction step increases uncertainty in the robot's belief.

3.2.4 Mathematical Derivation of the KF

This section derives the Kalman filter algorithm in Table 3.1. The section can safely be skipped at first reading.

Part 1: Prediction. Our derivation begins with Lines 2 and 3 of the algorithm, in which the belief $\overline{bel}(x_t)$ is calculated from the belief one time step earlier, $bel(x_{t-1})$. Lines 2 and 3 implement the update step described in Equation (2.61), restated here for the reader's convenience:

$$\overline{bel}(x_t) = \int \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\sim \mathcal{N}(x_t; A_t x_{t-1} + B_t u_t, R_t)} \underbrace{bel(x_{t-1})}_{\sim \mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} dx_{t-1} \tag{3.7}$$

The "prior" belief $bel(x_{t-1})$ is represented by the mean μ_{t-1} and the covariance Σ_{t-1}. The state transition probability $p(x_t \mid x_{t-1}, u_t)$ was given in (3.4) as a normal distribution over x_t with mean $A_t x_{t-1} + B_t u_t$ and covariance R_t. As we shall show now, the outcome of (3.7) is again a Gaussian with mean $\bar{\mu}_t$ and covariance $\bar{\Sigma}_t$ as stated in Table 3.1.

We begin by writing (3.7) in its Gaussian form:

$$\begin{aligned} \overline{bel}(x_t) &= \eta \int \exp\left\{-\tfrac{1}{2}\,(x_t - A_t\,x_{t-1} - B_t\,u_t)^T\,R_t^{-1}\,(x_t - A_t\,x_{t-1} - B_t\,u_t)\right\} \\ &\qquad \exp\left\{-\tfrac{1}{2}\,(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\}\,dx_{t-1}\,. \end{aligned} \tag{3.8}$$

In short, we have

$$\overline{bel}(x_t) = \eta \int \exp\{-L_t\}\,dx_{t-1} \tag{3.9}$$

with

$$\begin{aligned} L_t &= \tfrac{1}{2}\,(x_t - A_t\,x_{t-1} - B_t\,u_t)^T\,R_t^{-1}\,(x_t - A_t\,x_{t-1} - B_t\,u_t) \\ &\quad + \tfrac{1}{2}\,(x_{t-1} - \mu_{t-1})^T\,\Sigma_{t-1}^{-1}\,(x_{t-1} - \mu_{t-1}). \end{aligned} \tag{3.10}$$

Notice that L_t is quadratic in x_{t-1}; it is also quadratic in x_t.

Expression (3.9) contains an integral. To solve this integral in closed form, we will now decompose L_t into two functions, $L_t(x_{t-1}, x_t)$ and $L_t(x_t)$:

$$L_t = L_t(x_{t-1}, x_t) + L_t(x_t) \tag{3.11}$$

so that all terms containing x_{t-1} are collected in $L_t(x_{t-1}, x_t)$. This decomposition will allow us to move $L_t(x_t)$ outside the integration, since its value does not depend on the integration variable x_{t-1}:

$$\begin{aligned} \overline{bel}(x_t) &= \eta \int \exp\{-L_t\}\; dx_{t-1} \\ &= \eta \int \exp\{-L_t(x_{t-1}, x_t) - L_t(x_t)\}\; dx_{t-1} \\ &= \eta\,\exp\{-L_t(x_t)\} \int \exp\{-L_t(x_{t-1}, x_t)\}\; dx_{t-1} \end{aligned} \tag{3.12}$$

Furthermore, we will choose $L_t(x_{t-1}, x_t)$ such that the value of the integral in (3.12) does not depend on x_t. Thus, the integral will simply become a constant relative to the problem of estimating the belief distribution over x_t. The resulting distribution over x_t will be entirely defined through $L_t(x_t)$.

$$\overline{bel}(x_t) = \eta\,\exp\{-L_t(x_t)\} \tag{3.13}$$

Let us now perform this decomposition. We are seeking a function $L_t(x_{t-1}, x_t)$ quadratic in x_{t-1}. (This function will also depend on x_t, but that shall not concern us at this point.) To determine the coefficients of this quadratic, we calculate the first two derivatives of L_t:

$$\frac{\partial L_t}{\partial x_{t-1}} = -A_t^T\,R_t^{-1}\,(x_t - A_t\,x_{t-1} - B_t\,u_t) + \Sigma_{t-1}^{-1}\,(x_{t-1} - \mu_{t-1}) \tag{3.14}$$

$$\frac{\partial^2 L_t}{\partial x_{t-1}^2} \quad = \quad A_t^T \; R_t^{-1} \; A_t + \Sigma_{t-1}^{-1} \quad =: \quad \Psi_t^{-1} \tag{3.15}$$

Ψ_t defines the curvature of $L_t(x_{t-1}, x_t)$. Setting the first derivative of L_t to 0 gives us the mean:

$$A_t^T \; R_t^{-1} \; (x_t - A_t \; x_{t-1} - B_t \; u_t) \quad = \quad \Sigma_{t-1}^{-1} \; (x_{t-1} - \mu_{t-1}) \tag{3.16}$$

This expression is now solved for x_{t-1}

$$\begin{aligned}
&\iff \quad A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) - A_t^T \; R_t^{-1} \; A_t \; x_{t-1} \quad = \quad \Sigma_{t-1}^{-1} \; x_{t-1} - \Sigma_{t-1}^{-1} \; \mu_{t-1} \\
&\iff \quad A_t^T \; R_t^{-1} \; A_t \; x_{t-1} + \Sigma_{t-1}^{-1} \; x_{t-1} \quad = \quad A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1} \\
&\iff \quad (A_t^T \; R_t^{-1} \; A_t \; + \Sigma_{t-1}^{-1}) \; x_{t-1} \quad = \quad A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1} \\
&\iff \quad \Psi_t^{-1} \; x_{t-1} \quad = \quad A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1} \\
&\iff \quad x_{t-1} \quad = \quad \Psi_t \; [A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1}]
\end{aligned} \tag{3.17}$$

Thus, we now have a quadratic function $L_t(x_{t-1}, x_t)$, defined as follows:

$$\begin{aligned}
L_t(x_{t-1}, x_t) \quad = \quad & \tfrac{1}{2}(x_{t-1} - \Psi_t \; [A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1}])^T \; \Psi^{-1} \\
& (x_{t-1} - \Psi_t \; [A_t^T \; R_t^{-1} \; (x_t - B_t \; u_t) + \Sigma_{t-1}^{-1} \; \mu_{t-1}])
\end{aligned} \tag{3.18}$$

Clearly, this is not the only quadratic function satisfying our decomposition in (3.11). However, $L_t(x_{t-1}, x_t)$ is of the common quadratic form of the negative exponent of a normal distribution. In fact the function

$$\det(2\pi\Psi)^{-\frac{1}{2}} \; \exp\{-L_t(x_{t-1}, x_t)\} \tag{3.19}$$

is a valid probability density function (PDF) for the variable x_{t-1}. As the reader easily verifies, this function is of the form defined in (3.1). We know from (2.5) that PDFs integrate to 1. Thus, we have

$$\int \det(2\pi\Psi)^{-\frac{1}{2}} \; \exp\{-L_t(x_{t-1}, x_t)\} \; dx_{t-1} \quad = \quad 1 \tag{3.20}$$

From this it follows that

$$\int \exp\{-L_t(x_{t-1}, x_t)\}\, dx_{t-1} \quad = \quad \det(2\pi\Psi)^{\frac{1}{2}}\ . \tag{3.21}$$

The important thing to notice is that the value of this integral is *independent* of x_t, our target variable. Thus, for our problem of calculating a distribution over x_t, this integral is constant. Subsuming this constant into the normalizer η, we get the following expression for Equation (3.12):

$$\begin{aligned} \overline{bel}(x_t) &= \eta\ \exp\{-L_t(x_t)\} \int \exp\{-L_t(x_{t-1}, x_t)\}\ dx_{t-1} \\ &= \eta\ \exp\{-L_t(x_t)\} \end{aligned} \tag{3.22}$$

Notice that the normalizers η left and right of the equal sign are *not* the same. This decomposition establishes the correctness of (3.13).

It remains to determine the function $L_t(x_t)$, which is the difference of L_t, defined in (3.10), and $L_t(x_{t-1}, x_t)$, defined in (3.18):

$$\begin{aligned} L_t(x_t) &= L_t - L_t(x_{t-1}, x_t) \\ &= \tfrac{1}{2}\ (x_t - A_t\ x_{t-1} - B_t\ u_t)^T\ R_t^{-1}\ (x_t - A_t\ x_{t-1} - B_t\ u_t) \\ &\quad + \tfrac{1}{2}\ (x_{t-1} - \mu_{t-1})^T\ \Sigma_{t-1}^{-1}\ (x_{t-1} - \mu_{t-1}) \\ &\quad -\tfrac{1}{2}(x_{t-1} - \Psi_t\ [A_t^T\ R_t^{-1}\ (x_t - B_t\ u_t) + \Sigma_{t-1}^{-1}\ \mu_{t-1}])^T\ \Psi^{-1} \\ &\quad (x_{t-1} - \Psi_t\ [A_t^T\ R_t^{-1}\ (x_t - B_t\ u_t) + \Sigma_{t-1}^{-1}\ \mu_{t-1}]) \end{aligned} \tag{3.23}$$

Let us quickly verify that $L_t(x_t)$ indeed does not depend on x_{t-1}. To do so, we substitute back $\Psi_t = (A_t^T\ R_t^{-1}\ A_t + \Sigma_{t-1}^{-1})^{-1}$, and multiply out the terms above. For the reader's convenience, terms that contain x_{t-1} are underlined (doubly if they are quadratic in x_{t-1}).

$$\begin{aligned} L_t(x_t) &= \underline{\underline{\tfrac{1}{2}\ x_{t-1}^T A_t^T\ R_t^{-1}\ A_t\ x_{t-1}}}\ \underline{- x_{t-1}^T A_t^T\ R_t^{-1}\ (x_t - B_t\ u_t)} \\ &\qquad + \tfrac{1}{2}\ (x_t - B_t\ u_t)^T\ R_t^{-1}\ (x_t - B_t\ u_t) \\ &\quad \underline{\underline{+\tfrac{1}{2}\ x_{t-1}^T\ \Sigma_{t-1}^{-1}\ x_{t-1}}}\ \underline{- x_{t-1}^T\ \Sigma_{t-1}^{-1}\ \mu_{t-1}}\ + \tfrac{1}{2}\ \mu_{t-1}^T\ \Sigma_{t-1}^{-1}\ \mu_{t-1} \\ &\quad \underline{\underline{-\tfrac{1}{2}\ x_{t-1}^T\ (A_t^T\ R_t^{-1}\ A_t + \Sigma_{t-1}^{-1})\ x_{t-1}}} \end{aligned}$$

$$\begin{array}{l} \underline{+x_{t-1}^T\,[A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}]} \\ -\frac{1}{2}\;[A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}]^T\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1} \\ \qquad [A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}] \end{array} \tag{3.24}$$

It is now easily seen that all terms that contain x_{t-1} cancel out. This should come at no surprise, since it is a consequence of our construction of $L_t(x_{t-1}, x_t)$.

$$\begin{array}{rcl} L_t(x_t) & = & +\frac{1}{2}\;(x_t - B_t\;u_t)^T\;R_t^{-1}\;(x_t - B_t\;u_t)\;+\frac{1}{2}\;\mu_{t-1}^T\;\Sigma_{t-1}^{-1}\;\mu_{t-1} \\ & & -\frac{1}{2}\;[A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}]^T\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1} \\ & & \qquad [A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}] \end{array} \tag{3.25}$$

Furthermore, $L_t(x_t)$ is quadratic in x_t. This observation means that $\overline{bel}(x_t)$ is indeed normal distributed. The mean and covariance of this distribution are of course the minimum and curvature of $L_t(x_t)$, which we now easily obtain by by computing the first and second derivatives of $L_t(x_t)$ with respect to x_t:

$$\begin{array}{rcl} \frac{\partial L_t(x_t)}{\partial x_t} & = & R_t^{-1}\;(x_t - B_t\;u_t)\;\; - R_t^{-1}\;A_t\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1} \\ & & [A_t^T\;R_t^{-1}\;(x_t - B_t\;u_t) + \Sigma_{t-1}^{-1}\;\mu_{t-1}] \\ & = & [R_t^{-1} - R_t^{-1}\;A_t\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1} A_t^T\;R_t^{-1}]\;(x_t - B_t\;u_t) \\ & & - R_t^{-1}\;A_t\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1}\;\Sigma_{t-1}^{-1}\;\mu_{t-1} \end{array} \tag{3.26}$$

The *inversion lemma* stated (and proved) in Table 3.2 allows us to express the first factor as follows:

$$R_t^{-1} - R_t^{-1}\;A_t\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1} A_t^T\;R_t^{-1} \;\; = \;\; (R_t + A_t\;\Sigma_{t-1}\;A_t^T)^{-1} \tag{3.27}$$

Hence the desired derivative is given by the following expression:

$$\begin{array}{rcl} \frac{\partial L_t(x_t)}{\partial x_t} & = & (R_t + A_t\;\Sigma_{t-1}\;A_t^T)^{-1}\;(x_t - B_t\;u_t) \\ & & -R_t^{-1}\;A_t\;(A_t^T\;R_t^{-1}\;A_t + \Sigma_{t-1}^{-1})^{-1}\;\Sigma_{t-1}^{-1}\;\mu_{t-1} \end{array} \tag{3.28}$$

Inversion Lemma. For any invertible quadratic matrices R and Q and any matrix P with appropriate dimension, the following holds true

$$(R + P\,Q\,P^T)^{-1} \;=\; R^{-1} - R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1}$$

assuming that all above matrices can be inverted as stated.

Proof. It suffices to show that

$$(R^{-1} - R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1})\,(R + P\,Q\,P^T) \;=\; I$$

This is shown through a series of transformations:

$$\begin{aligned}
&= \underbrace{R^{-1}\,R}_{=I} + R^{-1}\,P\,Q\,P^T \;-\; R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,\underbrace{R^{-1}\,R}_{=I} \\
&\quad - R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1}\,P\,Q\,P^T \\
&= I \;+\; R^{-1}\,P\,Q\,P^T \;-\; R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T \\
&\quad - R^{-1}\,P\,(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1}\,P\,Q\,P^T \\
&= I \;+\; R^{-1}\,P\,[Q\,P^T \;-\; (Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T \\
&\quad - (Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1}\,P\,Q\,P^T] \\
&= I \;+\; R^{-1}\,P\,[Q\,P^T \;-\; (Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,\underbrace{Q^{-1}\,Q}_{=I}\,P^T \\
&\quad - (Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,P^T\,R^{-1}\,P\,Q\,P^T] \\
&= I \;+\; R^{-1}\,P\,[Q\,P^T \;-\; \underbrace{(Q^{-1} + P^T\,R^{-1}\,P)^{-1}\,(Q^{-1} + P^T\,R^{-1}\,P)}_{=I}\,Q\,P^T] \\
&= I \;+\; R^{-1}\,P\,[\underbrace{Q\,P^T \;-\; Q\,P^T}_{=0}] \;=\; I
\end{aligned}$$

Table 3.2 The (specialized) inversion lemma.

The minimum of $L_t(x_t)$ is attained when the first derivative is zero.

$$(R_t + A_t\,\Sigma_{t-1}\,A_t^T)^{-1}\,(x_t - B_t\,u_t) \;=\; R_t^{-1}\,A_t\,(A_t^T\,R_t^{-1}\,A_t + \Sigma_{t-1}^{-1})^{-1}\,\Sigma_{t-1}^{-1}\,\mu_{t-1} \tag{3.29}$$

Solving this for the target variable x_t gives us the surprisingly compact result

$$\begin{aligned} x_t &= B_t\ u_t + \underbrace{(R_t + A_t\ \Sigma_{t-1}\ A_t^T) R_t^{-1}\ A_t}_{A_t + A_t\ \Sigma_{t-1}\ A_t^T R_t^{-1}\ A_t} \underbrace{(A_t^T\ R_t^{-1}\ A_t + \Sigma_{t-1}^{-1})^{-1}\ \Sigma_{t-1}^{-1}}_{(\Sigma_{t-1} A_t^T\ R_t^{-1}\ A_t + I)^{-1}}\ \mu_{t-1} \\ &= B_t\ u_t + A_t\ \underbrace{(I + \Sigma_{t-1}\ A_t^T R_t^{-1}\ A_t)\ (\Sigma_{t-1} A_t^T\ R_t^{-1}\ A_t + I)^{-1}}_{= I}\ \mu_{t-1} \\ &= B_t\ u_t + A_t\ \mu_{t-1} \end{aligned} \tag{3.30}$$

Thus, the mean of the belief $\overline{bel}(x_t)$ after incorporating the motion command u_t is $B_t\ u_t + A_t\ \mu_{t-1}$. This proves the correctness of Line 2 of the Kalman filter algorithm in Table 3.1. Line 3 is now obtained by calculating the second derivative of $L_t(x_t)$:

$$\frac{\partial^2 L_t(x_t)}{\partial x_t^2} = (R_t + A_t\ \Sigma_{t-1}\ A_t^T)^{-1} \tag{3.31}$$

This is the curvature of the quadratic function $L_t(x_t)$, whose inverse is the covariance of the belief $\overline{bel}(x_t)$.

To summarize, we showed that the prediction steps in Lines 2 and 3 of the Kalman filter algorithm indeed implement the Bayes filter prediction step. To do so, we first decomposed the exponent of the belief $\overline{bel}(x_t)$ into two functions, $L_t(x_{t-1}, x_t)$ and $L_t(x_t)$. Then we showed that $L_t(x_{t-1}, x_t)$ changes the predicted belief $\overline{bel}(x_t)$ only by a constant factor, which can be subsumed into the normalizing constant η. Finally, we determined the function $L_t(x_t)$ and showed that it results in the mean $\bar{\mu}_t$ and covariance $\bar{\Sigma}_t$ of the Kalman filter prediction $\overline{bel}(x_t)$.

Part 2: Measurement Update. We will now derive the measurement update in Lines 4, 5, and 6 (Table 3.1) of our Kalman filter algorithm. We begin with the general Bayes filter mechanism for incorporating measurements, stated in Equation (2.58) and restated here in annotated form:

$$bel(x_t) = \eta\ \underbrace{p(z_t \mid x_t)}_{\sim \mathcal{N}(z_t; C_t x_t, Q_t)}\ \underbrace{\overline{bel}(x_t)}_{\sim \mathcal{N}(x_t; \bar{\mu}_t, \bar{\Sigma}_t)} \tag{3.32}$$

The mean and covariance of $\overline{bel}(x_t)$ are obviously given by $\bar{\mu}_t$ and $\bar{\Sigma}_t$. The measurement probability $p(z_t \mid x_t)$ was defined in (3.6) to be normal as well, with mean $C_t\ x_t$

and covariance Q_t. Thus, the product is given by an exponential

$$bel(x_t) \quad = \quad \eta \ \exp\{-J_t\} \tag{3.33}$$

with

$$J_t \quad = \quad \tfrac{1}{2} \ (z_t - C_t x_t)^T \ Q_t^{-1} \ (z_t - C_t x_t) + \tfrac{1}{2} \ (x_t - \bar{\mu}_t)^T \ \bar{\Sigma}_t^{-1} \ (x_t - \bar{\mu}_t) \tag{3.34}$$

This function is quadratic in x_t, hence $bel(x_t)$ is a Gaussian. To calculate its parameters, we once again calculate the first two derivatives of J_t with respect to x_t:

$$\frac{\partial J}{\partial x_t} \quad = \quad -C_t^T \ Q_t^{-1} \ (z_t - C_t x_t) + \bar{\Sigma}_t^{-1} \ (x_t - \bar{\mu}_t) \tag{3.35}$$

$$\frac{\partial^2 J}{\partial x_t^2} \quad = \quad C_t^T \ Q_t^{-1} \ C_t + \bar{\Sigma}_t^{-1} \tag{3.36}$$

The second term is the inverse of the covariance of $bel(x_t)$:

$$\Sigma_t \quad = \quad (C_t^T \ Q_t^{-1} \ C_t + \bar{\Sigma}_t^{-1})^{-1} \tag{3.37}$$

The mean of $bel(x_t)$ is the minimum of this quadratic function, which we now calculate by setting the first derivative of J_t to zero (and substituting μ_t for x_t):

$$C_t^T \ Q_t^{-1} \ (z_t - C_t \mu_t) \quad = \quad \bar{\Sigma}_t^{-1} \ (\mu_t - \bar{\mu}_t) \tag{3.38}$$

The expression on the left of the equal sign can be transformed as follows:

$$\begin{aligned} & C_t^T \ Q_t^{-1} \ (z_t - C_t \mu_t) \\ & \quad = \quad C_t^T \ Q_t^{-1} \ (z_t - C_t \ \mu_t + C_t \ \bar{\mu}_t - C_t \ \bar{\mu}_t) \\ & \quad = \quad C_t^T \ Q_t^{-1} \ (z_t - C_t \bar{\mu}_t) - C_t^T \ Q_t^{-1} \ C_t \ (\mu_t - \bar{\mu}_t) \end{aligned} \tag{3.39}$$

Substituting this back into (3.38) gives us

$$C_t^T \ Q_t^{-1} \ (z_t - C_t \bar{\mu}_t) \quad = \quad \underbrace{(C_t^T \ Q_t^{-1} \ C_t + \bar{\Sigma}_t^{-1})}_{= \Sigma_t^{-1}} \ (\mu_t - \bar{\mu}_t) \tag{3.40}$$

and hence we have

$$\Sigma_t \, C_t^T \, Q_t^{-1} \, (z_t - C_t \, \bar{\mu}_t) \quad = \quad \mu_t - \bar{\mu}_t \tag{3.41}$$

We now define the Kalman gain as

$$K_t \quad = \quad \Sigma_t \, C_t^T \, Q_t^{-1} \tag{3.42}$$

and obtain

$$\mu_t \quad = \quad \bar{\mu}_t + K_t \, (z_t - C_t \, \bar{\mu}_t) \tag{3.43}$$

This proves the correctness of Line 5 in the Kalman filter algorithm in Table 3.1.

The Kalman gain, as defined in (3.42), is a function of Σ_t. This is at odds with the fact that we utilize K_t to calculate Σ_t in Line 6 of the algorithm. The following transformation shows us how to expresses K_t in terms of covariances other than Σ_t. It begins with the definition of K_t in (3.42):

$$\begin{aligned}
K_t &= \Sigma_t \, C_t^T \, Q_t^{-1} \\
&= \Sigma_t \, C_t^T \, Q_t^{-1} \, \underbrace{(C_t \, \bar{\Sigma}_t \, C_t^T + Q_t) \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}}_{= I} \\
&= \Sigma_t \, (C_t^T \, Q_t^{-1} \, C_t \, \bar{\Sigma}_t \, C_t^T + C_t^T \, \underbrace{Q_t^{-1} \, Q_t}_{= I}) \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1} \\
&= \Sigma_t \, (C_t^T \, Q_t^{-1} \, C_t \, \bar{\Sigma}_t \, C_t^T + C_t^T) \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1} \\
&= \Sigma_t \, (C_t^T \, Q_t^{-1} \, C_t \, \bar{\Sigma}_t \, C_t^T + \underbrace{\bar{\Sigma}_t^{-1} \, \bar{\Sigma}_t}_{= I} \, C_t^T) \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1} \\
&= \Sigma_t \, \underbrace{(C_t^T \, Q_t^{-1} \, C_t + \bar{\Sigma}_t^{-1})}_{= \Sigma_t^{-1}} \, \bar{\Sigma}_t \, C_t^T \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1} \\
&= \underbrace{\Sigma_t \, \Sigma_t^{-1}}_{= I} \, \bar{\Sigma}_t \, C_t^T \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1} \\
&= \bar{\Sigma}_t \, C_t^T \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}
\end{aligned} \tag{3.44}$$

This expression proves the correctness of Line 4 of our Kalman filter algorithm. Line 6 is obtained by expressing the covariance using the Kalman gain K_t. The advantage

of the calculation in Table 3.1 over the definition in Equation (3.37) lies in the fact that we can avoid inverting the state covariance matrix. This is essential for applications of Kalman filters to high-dimensional state spaces.

Our transformation is once again carried out using the *inversion lemma*, which was already stated in Table 3.2. Here we restate it using the notation of Equation (3.37):

$$(\bar{\Sigma}_t^{-1} + C_t^T\ Q_t^{-1}\ C_t)^{-1} \quad = \quad \bar{\Sigma}_t - \bar{\Sigma}_t\ C_t^T\ (Q_t + C_t\ \bar{\Sigma}_t\ C_t^T)^{-1}\ C_t\ \bar{\Sigma}_t \tag{3.45}$$

This lets us arrive at the following expression for the covariance:

$$\begin{aligned}
\Sigma_t &= (C_t^T\ Q_t^{-1}\ C_t + \bar{\Sigma}_t^{-1})^{-1} \\
&= \bar{\Sigma}_t - \bar{\Sigma}_t\ C_t^T\ (Q_t + C_t\ \bar{\Sigma}_t\ C_t^T)^{-1}\ C_t\ \bar{\Sigma}_t \\
&= [I - \underbrace{\bar{\Sigma}_t\ C_t^T\ (Q_t + C_t\ \bar{\Sigma}_t\ C_t^T)^{-1}}_{= K_t, \text{ see Eq. (3.44)}}\ C_t]\ \bar{\Sigma}_t \\
&= (I - K_t\ C_t)\ \bar{\Sigma}_t
\end{aligned} \tag{3.46}$$

This proves the correctness of Line 6 of our Kalman filter algorithm.

3.3 THE EXTENDED KALMAN FILTER

The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely fulfilled in practice. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear next state transitions. This observation, along with the assumption of unimodal beliefs, renders plain Kalman filters, as discussed so far, inapplicable to all but the most trivial robotics problems.

The extended Kalman filter (EKF) overcomes one of these assumptions: the linearity assumption. Here the assumption is that the next state probability and the measurement probabilities are governed by nonlinear functions g and h, respectively:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3.47}$$

$$z_t = h(x_t) + \delta_t\ . \tag{3.48}$$

This model strictly generalizes the linear Gaussian model underlying Kalman filters, postulated in Equations (3.2) and (3.5). The function g replaces the matrices A_t and B_t in (3.2), and h replaces the matrix C_t in (3.5). Unfortunately, with arbitrary functions g and h, the belief is no longer a Gaussian. In fact, performing the belief update exactly is usually impossible for nonlinear functions g and h, in the sense that the Bayes filter does not possess a closed-form solution.

The extended Kalman filter (EKF) calculates an approximation to the true belief. It represents this approximation by a Gaussian. In particular, the belief $bel(x_t)$ at time t is represented by a mean μ_t and a covariance Σ_t. Thus, the EKF inherits from the Kalman filter the basic belief representation, but it differs in that this belief is only approximate, not exact as was the case in Kalman filters.

3.3.1 Linearization Via Taylor Expansion

The key idea underlying the EKF is called *linearization*. Figure **??** illustrates the basic concept. Suppose we are given a nonlinear next state function g. A Gaussian projected through this function is typically non-Gaussian. This is because nonlinearities in g distort the belief in ways that destroys its nice Gaussian shape, as illustrated in the figure. Linearization approximates g by a linear function that is tangent to g at the mean of the Gaussian. By projecting the Gaussian through this linear approximation, the posterior is Gaussian. In fact, once g is linearized, the mechanics of belief propagation are equivalent to those of the Kalman filter. The same argument applies to the multiplication of Gaussians when a measurement function h is involved. Again, the EKF approximates h by a linear function tangent to h, thereby retaining the Gaussian nature of the posterior belief.

There exist many techniques for linearizing nonlinear functions. EKFs utilize a method called (first order) *Taylor expansion*. Taylor expansion construct a linear approximation to a function g from g's value and slope. The slope is given by the partial derivative

$$g'(u_t, x_{t-1}) \quad := \quad \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{3.49}$$

Clearly, both the value of g and its slope depend on the argument of g. A logical choice for selecting the argument is to chose the state deemed most likely at the time of linearization. For Gaussians, the most likely state is the mean of the posterior μ_{t-1}. In other words, g is approximated by its value at μ_{t-1} (and at u_t), and the linear

extrapolation is achieved by a term proportional to the gradient of g at μ_{t-1} and u_t:

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=: G_t} (x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1}) \end{aligned} \tag{3.50}$$

Written as Gaussian, the next state probability is approximated as follows:

$$\begin{aligned} & p(x_t \mid u_t, x_{t-1}) \\ & \quad \approx \det (2\pi R_t)^{-\frac{1}{2}} \exp \left\{ -\tfrac{1}{2} \, [x_t - g(u_t, \mu_{t-1}) - G_t \, (x_{t-1} - \mu_{t-1})]^T \right. \\ & \qquad \left. R_t^{-1} \, [x_t - g(u_t, \mu_{t-1}) - G_t \, (x_{t-1} - \mu_{t-1})] \right\} \end{aligned} \tag{3.51}$$

Notice that G_t is a matrix of size $n \times n$, with n denoting the dimension of the state. This matrix is often called the *Jacobian*. The value of the Jacobian depends on u_t and μ_{t-1}, hence it differs for different points in time.

EKFs implement the exact same linearization for the measurement function h. Here the Taylor expansion is developed around $\bar{\mu}_t$, the state deemed most likely by the robot at the time it linearizes h:

$$\begin{aligned} h(x_t) &\approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{=: H_t} (x_t - \bar{\mu}_t) \\ &= h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t) \end{aligned} \tag{3.52}$$

with $h'(x_t) = \frac{\partial h(x_t)}{\partial x_t}$. Written as a Gaussian, we have

$$\begin{aligned} p(z_t \mid x_t) = & \det (2\pi Q_t)^{-\frac{1}{2}} \exp \left\{ -\tfrac{1}{2} \, [z_t - h(\bar{\mu}_t) - H_t \, (x_t - \bar{\mu}_t)]^T \right. \\ & \left. Q_t^{-1} \, [z_t - h(\bar{\mu}_t) - H_t \, (x_t - \bar{\mu}_t)] \right\} \end{aligned} \tag{3.53}$$

3.3.2 The EKF Algorithm

Table 3.3 states the EKF algorithm. In many ways, this algorithm is similar to the Kalman filter algorithm stated in Table 3.1. The most important differences are summarized by the following table:

1: **Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):**
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \; \Sigma_{t-1} \; G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t \; H_t^T (H_t \; \bar{\Sigma}_t \; H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t \; H_t) \; \bar{\Sigma}_t$
7: return μ_t, Σ_t

Table 3.3 The extended Kalman filter (EKF) algorithm.

	Kalman filter	EKF
state prediction (Line 2)	$A_t \; \mu_{t-1} + B_t \; u_t$	$g(u_t, \mu_{t-1})$
measurement prediction (Line 5)	$C_t \; \bar{\mu}_t$	$h(\bar{\mu}_t)$

That is, the linear predictions in Kalman filters are replaced by their nonlinear generalizations in EKFs. Moreover, EKFs use Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t, and C_t in Kalman filters. The Jacobian G_t corresponds to the matrices A_t and B_t, and the Jacobian H_t corresponds to C_t. A detailed example for extended Kalman filters will be given in Chapter **??**.

3.3.3 Mathematical Derivation of the EKF

The mathematical derivation of the EKF parallels that of the Kalman filter in Section 3.2.4, and hence shall only be sketched here. The prediction is calculated as follows (cf. (3.7)):

$$\overline{bel}(x_t) \quad = \quad \int \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\sim \mathcal{N}(x_t; g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}), R_t)} \quad \underbrace{bel(x_{t-1})}_{\sim \mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} \quad dx_{t-1} \tag{3.54}$$

This distribution is the EKF analog of the prediction distribution in the Kalman filter, stated in (3.7). The Gaussian $p(x_t \mid x_{t-1}, u_t)$ can be found in Equation (3.51). The

function L_t is given by (cf. (3.10))

$$\begin{aligned} L_t &= \tfrac{1}{2}\,(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1}))^T \\ &\quad R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})) \\ &\quad + \tfrac{1}{2}\,(x_{t-1} - \mu_{t-1})^T\,\Sigma_{t-1}^{-1}\,(x_{t-1} - \mu_{t-1}) \end{aligned} \tag{3.55}$$

which is quadratic in both x_{t-1} and x_t, as above. As in (3.11), we decompose L_t into $L_t(x_{t-1}, x_t)$ and $L_t(x_t)$:

$$\begin{aligned} &L_t(x_{t-1}, x_t) \\ &\quad = \tfrac{1}{2}\,(x_{t-1} - \Phi_t\,[G_t^T\,R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}])^T\,\Phi^{-1} \\ &\qquad (x_{t-1} - \Phi_t\,[G_t^T\,R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}]) \end{aligned} \tag{3.56}$$

with

$$\Phi_t = (G_t^T\,R_t^{-1}\,G_t + \Sigma_{t-1}^{-1})^{-1} \tag{3.57}$$

and hence

$$\begin{aligned} L_t(x_t) &= \tfrac{1}{2}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1})^T\,R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) \\ &\quad + \tfrac{1}{2}\,(x_{t-1} - \mu_{t-1})^T\,\Sigma_{t-1}^{-1}\,(x_{t-1} - \mu_{t-1}) \\ &\quad - \tfrac{1}{2}\,[G_t^T\,R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}]^T \\ &\quad \Phi_t\,[G_t^T\,R_t^{-1}\,(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}] \end{aligned} \tag{3.58}$$

As the reader easily verifies, setting the first derivative of $L_t(x_t)$ to zero gives us the update $\mu_t = g(u_t, \mu_{t-1})$, in analogy to the derivation in Equations (3.26) through (3.30). The second derivative is given by $(R_t + G_t\,\Sigma_{t-1}\,G_t^T)^{-1}$ (see (3.31)).

The measurement update is also derived analogously to the Kalman filter in Section 3.2.4. In analogy to (3.32), we have for the EKF

$$bel(x_t) = \eta \underbrace{p(z_t \mid x_t)}_{\sim\mathcal{N}(z_t; h(\bar\mu_t) + H_t\,(x_t - \bar\mu_t), Q_t)} \underbrace{\overline{bel}(x_t)}_{\sim\mathcal{N}(x_t; \bar\mu_t, \bar\Sigma_t)} \tag{3.59}$$

using the linearized next state transition function from (3.52). This leads to the exponent (see (3.34)):

$$\begin{aligned} J_t &= \tfrac{1}{2}\,(z_t - h(\bar{\mu}_t) - H_t\,(x_t - \bar{\mu}_t))^T\,Q_t^{-1}\,(z_t - h(\bar{\mu}_t) - H_t\,(x_t - \bar{\mu}_t)) \\ &\quad + \tfrac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t) \end{aligned} \tag{3.60}$$

The resulting mean and covariance is given by

$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \tag{3.61}$$

$$\Sigma_t = (I - K_t\,H_t)\,\bar{\Sigma}_t \tag{3.62}$$

with the Kalman gain

$$K_t = \bar{\Sigma}_t\,H_t^T\,(H_t\,\bar{\Sigma}_{t-1}\,H_t^T + Q_t)^{-1} \tag{3.63}$$

The derivation of these equations is analogous to Equations (3.35) through (3.46).

3.3.4 Practical Considerations

The EKF has become just about the most popular tool for state estimation in robotics. Its strength lies in its simplicity and in its computational efficiency. As was the case for the Kalman filter, each update requires time $O(k^{2.8} + n^2)$, where k is the dimension of the measurement vector z_t, and n is the dimension of the state vector x_t. Other algorithms, such as the particle filter discussed further below, may require time exponential in n.

The EKF owes its computational efficiency to the fact that it represents the belief by a multivariate Gaussian distribution. A Gaussian is a unimodal distribution, which can be thought of as a single guess, annotated with an uncertainty ellipse. In many practical problems, Gaussians are robust estimators. Applications of the Kalman filter to state spaces with 1,000 dimensions or more will be discussed in later chapters of this book. EKFs have been applied with great success to a number of state estimation problems that violate the underlying assumptions.

Sometimes, one might want to pursue multiple distinct hypotheses. For example, a robot might have two distinct hypotheses as to where it is, but the arithmetic mean of

these hypotheses is not a likely contender. Such situations require multi-modal representations for the posterior belief. EKFs, in the form described here, are incapable of representing such multimodal beliefs. A common extension of EKFs is to represent posteriors using *mixtures*, or *sums*, of Gaussians. A mixture of J Gaussians may be of the form (cf. (**??**)):

$$bel(x) \quad = \quad \sum_j a_j \ \det\left(2\pi\Sigma_{j,t}\right)^{-\frac{1}{2}} \ \exp\left\{-\tfrac{1}{2}(x_t - \mu_{j,t})^T \ \Sigma_{j,t}^{-1} \ (x_t - \mu_{j,t})\right\} \tag{3.64}$$

where a_j are mixture parameters with $a_j \geq 0$ and $\sum_j a_j = 1$. EKFs that utilize such mixture representations are called *multi-hypothesis (extended) Kalman filters*, of MHEKF.

An important limitation of the EKF arises from the fact that it approximates state transitions and measurements using linear Taylor expansions. In virtually all robotics problems, these functions are nonlinear. The goodness of this approximation depends on two main factors. First, it depends on the degree of nonlinearity of the functions that are being approximated. If these functions are approximately linear, the EKF approximation may generally be a good one, and EKFs may approximate the posterior belief with sufficient accuracy. However, sometimes, the functions are not only nonlinear, but are also multi-modal, in which case the linearization may be a poor approximation. The goodness of the linearization also depends on the degree of uncertainty. The less certain the robot, the wider its Gaussian belief, and the more it is affected by nonlinearities in the state transition and measurement functions. In practice, when applying EKFs it is therefore important to keep the uncertainty of the state estimate small.

We also note that Taylor series expansion is only one way to linearize. Two other approaches have often been found to yield superior results. One is the *unscented Kalman filter*, which probes the function to be linearized at selected points and calculates a linearized approximation based on the outcomes of these probes. Another is known as *moments matching*, in which the linearization is calculated in a way that preserves the true mean and the true covariance of the posterior distribution (which is not the case for EKFs). Both techniques are relatively recent but appear to be superior to the EKF linearization.

3.4 THE INFORMATION FILTER

The dual of the Kalman filter is the information filter. Just like the KF and its nonlinear version, the EKF, the information filter (IF) represents the belief by a Gaussian. Thus, the standard information filter is subject to the same assumptions underlying the Kalman filter. The key difference between the KF and the IF arises from the way the Gaussian belief is represented. Whereas in the Kalman filter family of algorithms, Gaussians are represented by their moments (mean, covariance), information filters represent Gaussians in their canonical representation, which is comprised of an information matrix and an information vector. The difference in representation leads to different update equations. In particular, what is computationally complex in one representation happens to be simple in the other (and vice versa). The canonical and the moments representations are often considered *dual* to each other, and thus are the IF and the KF.

3.4.1 Canonical Representation

The canonical representation of a multivariate Gaussian is given by a matrix Ω and a vector ξ. The matrix Ω is the inverse of the covariance matrix:

$$\Omega = \Sigma^{-1} . \tag{3.65}$$

Ω is called the *information matrix*, or sometimes the *precision matrix*. The vector ξ is called the *information vector*. It is defined as

$$\xi = \Sigma^{-1} \mu . \tag{3.66}$$

It is easy to see that Ω and ξ are a complete parameterization of a Gaussian. In particular, the mean and covariance of the Gaussian can easily be obtained from the canonical representation by the inverse of (3.65) and (3.66):

$$\Sigma = \Omega^{-1} \tag{3.67}$$

$$\mu = \Omega^{-1} \xi \tag{3.68}$$

The canonical representation is often derived by multiplying out the exponent of a Gaussian. In (3.1), we defined the multivariate normal distribution as follows:

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{3.69}$$

A straightforward sequence of transformations leads to the following parameterization:

$$\begin{aligned} p(x) &= \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu - \tfrac{1}{2}\mu^T\Sigma^{-1}\mu\right\} \\ &= \underbrace{\det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}\mu^T\Sigma^{-1}\mu\right\}}_{\text{const.}} \exp\left\{-\tfrac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu\right\} \end{aligned} \tag{3.70}$$

The term labeled "const." does not depend on the target variable x. Hence, it can be subsumed into the normalizer η.

$$p(x) = \eta \exp\left\{-\tfrac{1}{2}x^T\ \Sigma^{-1}\ x + x^T\ \Sigma^{-1}\ \mu\right\} \tag{3.71}$$

This form motivates the parameterization of a Gaussian by its canonical parameters Ω and ξ.

$$p(x) = \eta \exp\left\{-\tfrac{1}{2}x^T\ \Omega\ x + x^T\ \xi\right\} \tag{3.72}$$

In many ways, the canonical representation is more elegant than the moments representation. In particular, the negative logarithm of the Gaussian (which plays an essential role in information theory) is a quadratic function in the canonical parameters Ω and ξ:

$$-\log p(x) = \text{const.} + \tfrac{1}{2}x^T\ \Omega\ x - x^T\ \xi \tag{3.73}$$

Here "const." is a constant. The reader may notice that we cannot use the symbol η to denote this constant, since negative logarithms of probabilities do not normalize to 1. The negative logarithm of our distribution $p(x)$ is quadratic in x, with the quadratic term parameterized by Ω and the linear term by ξ. In fact, for Gaussians, Ω must

1: **Algorithm Information_filter($\xi_{t-1}, \Omega_{t-1}, u_t, z_t$):**
2: $\bar{\Omega}_t = (A_t\ \Omega_{t-1}^{-1}\ A_t^T + R_t)^{-1}$
3: $\bar{\xi}_t = \bar{\Omega}_t(A_t\ \Omega_{t-1}^{-1}\ \xi_{t-1} + B_t\ u_t)$
4: $\Omega_t = C_t^T\ Q_t^{-1}\ C_t + \bar{\Omega}_t$
5: $\xi_t = C_t^T\ Q_t^{-1}\ z_t + \bar{\xi}_t$
6: return ξ_t, Ω_t

Table 3.4 The information filter (IF) algorithm.

be positive semidefinite, hence $-\log p(x)$ is a quadratic distance function with mean $\mu = \Omega^{-1}\ \xi$. This is easily verified by setting the first derivative of (3.73) to zero:

$$\frac{\partial[-\log p(x)]}{\partial x} \;=\; 0 \;\iff\; \Omega x - \xi \;=\; 0 \;\iff\; x = \Omega^{-1}\xi \tag{3.74}$$

The matrix Ω determines the rate at which the distance function increases in the different dimensions of the variable x. A quadratic distance that is weighted by a matrix Ω is called *Mahalanobis distance*.

3.4.2 The Information Filter Algorithm

Table 3.4 states the update algorithm known as information filter. Its input is a Gaussian in its canonical representation ξ_{t-1} and Ω_{t-1}, representing the belief at time $t-1$. Just like all Bayes filters, its input includes the control u_t and the measurement z_t. The output are the parameters ξ_t and Ω_t of the updated Gaussian.

The update involves matrices A_t, B_t, C_t, R_t, and Q_t. Those were defined in Section 3.2. The information filter assumes that the state transition and measurement probabilities are governed by the following linear Gaussian equations, originally defined in (3.2) and (3.5):

$$x_t \;=\; A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{3.75}$$

$$z_t \;=\; C_t x_t + \delta_t \tag{3.76}$$

Here R_t and Q_t are the covariances of the zero-mean noise variables ε_t and δ_t, respectively.

Just like the Kalman filter, the information filter is updated in two steps, a prediction step and a measurement update step. The prediction step is implemented in Lines 2 and 3 in Table 3.4. The parameters $\bar{\xi}_t$ and $\bar{\Omega}_t$ describe the Gaussian belief over x_t after incorporating the control u_t, but before incorporating the measurement z_t. The latter is done through Lines 4 and 5. Here the belief is updated based on the measurement z_t.

These two update steps can be vastly different in complexity, especially if the state space possesses many dimensions. The prediction step, as stated in Table 3.4, involves the inversion of two matrices of the size $n \times n$, where n is the dimension of the state space. This inversion requires approximately $O(n^{2.8})$ time. In Kalman filters, the update step is additive and requires at most $O(n^2)$ time; it requires less time if only a subset of variables is affected by a control, or if variables transition independently of each other. These roles are reversed for the measurement update step. Measurement updates are additive in the information filter. They require at most $O(n^2)$ time, and they are even more efficient if measurements carry only information about a subset of all state variables at a time. The measurement update is the difficult step in Kalman filters. It requires matrix inversion whose worst case complexity is $O(n^{2.8})$. This illustrates the dual character of Kalman and information filters.

3.4.3 Mathematical Derivation of the Information Filter

The derivation of the information filter is analogous to that of the Kalman filter. To derive the prediction step (Lines 2 and 3 in Table 3.4), we begin with the corresponding update equations of the Kalman filters, which can be found in Lines 2 and 3 of the algorithm in Table 3.1 and are restated here for the reader's convenience:

$$\bar{\mu}_t = A_t \, \mu_{t-1} + B_t \, u_t \tag{3.77}$$

$$\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t \tag{3.78}$$

The information filter prediction step follows now directly by substituting the moments μ and Σ by the canonical parameters ξ and Ω according to their definitions in

(3.67) and (3.72):

$$\begin{aligned} \mu_{t-1} &= \Omega_{t-1}^{-1}\ \xi_{t-1} \\ \Sigma_{t-1} &= \Omega_{t-1}^{-1} \end{aligned} \tag{3.79}$$

Substituting these expressions in (3.77) and (3.78) gives us the set of prediction equations

$$\bar{\Omega}_t = (A_t\ \Omega_{t-1}^{-1}\ A_t^T + R_t)^{-1} \tag{3.80}$$

$$\bar{\xi}_t = \bar{\Omega}_t(A_t\ \Omega_{t-1}^{-1}\ \xi_{t-1} + B_t\ u_t) \tag{3.81}$$

These equations are identical to those in Table 3.4. As is easily seen, the prediction step involves two nested inversions of a potentially large matrix. These nested inversions can be avoided when only a small number of state variables is affected by the motion update, a topic which will be discussed later in this book.

The derivation of the measurement update is even simpler. We begin with the Gaussian of the belief at time t, which was provided in Equation (3.34) and is restated here once again:

$$\begin{aligned} & bel(x_t) \\ &= \eta\ \exp\left\{-\tfrac{1}{2}\ (z_t - C_t x_t)^T\ Q_t^{-1}\ (z_t - C_t x_t) - \tfrac{1}{2}\ (x_t - \bar{\mu}_t)^T\ \bar{\Sigma}_t^{-1}\ (x_t - \bar{\mu}_t)\right\} \end{aligned} \tag{3.82}$$

For Gaussians represented in their canonical form this distribution is given by

$$\begin{aligned} & bel(x_t) \\ &= \eta\ \exp\left\{-\tfrac{1}{2}\ x_t^T\ C_t^T\ Q_t^{-1}\ C_t\ x_t + x_t^T\ C_t^T\ Q_t^{-1}\ z_t - \tfrac{1}{2}\ x_t^T\ \bar{\Omega}_t x_t + x_t^T \bar{\xi}_t\right\} \end{aligned} \tag{3.83}$$

which by reordering the terms in the exponent resolves to

$$bel(x_t) = \eta\ \exp\left\{-\tfrac{1}{2}\ x_t^T\ [C_t^T\ Q_t^{-1}\ C_t + \bar{\Omega}_t]\ x_t + x_t^T\ [C_t^T\ Q_t^{-1}\ z_t + \bar{\xi}_t]\right\}$$

We can now read off the measurement update equations, by collecting the terms in the squared brackets:

$$\xi_t = C_t^T\ Q_t^{-1}\ z_t + \bar{\xi}_t \tag{3.84}$$

1: **Algorithm Extended_information_filter($\xi_{t-1}, \Omega_{t-1}, u_t, z_t$):**
2: $\mu_{t-1} = \Omega_{t-1}^{-1}\ \xi_{t-1}$
3: $\bar{\Omega}_t = (G_t\ \Omega_{t-1}^{-1}\ G_t^T + R_t)^{-1}$
4: $\bar{\xi}_t = \bar{\Omega}_t\ g(u_t, \mu_{t-1})$
5: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
6: $\Omega_t = \bar{\Omega}_t + H_t^T\ Q_t^{-1}\ H_t$
7: $\xi_t = \bar{\xi}_t + H_t^T\ Q_t^{-1}\ [z_t - h(\bar{\mu}_t) - H_t\ \bar{\mu}_t]$
8: return ξ_t, Ω_t

Table 3.5 The extended information filter (EIF) algorithm.

$$\Omega_t = C_t^T\ Q_t^{-1}\ C_t + \bar{\Omega}_t \tag{3.85}$$

These equations are identical to the measurement update equations in Lines 4 and 5 of Table 3.4.

3.4.4 The Extended Information Filter Algorithm

The extended version of the information filter is analog to the EKF. Table 3.5 depicts the algorithm. The prediction is realized in Lines 2 through 4, and the measurement update in Lines 5 through 7. These update equations are largely analog to the linear information filter, with the functions g and h (and their Jacobian G_t and H_t) replacing the parameters of the linear model A_t, B_t, and C_t. As before, g and h specify the nonlinear next state function and measurement function, respectively. Those were defined in (3.47) and (3.48) and are restated here:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3.86}$$
$$z_t = h(x_t) + \delta_t\ . \tag{3.87}$$

Unfortunately, both g and h require a state as an input. This mandates the recovery of a state estimate μ from the canonical parameters. The recovery takes place in Line 2, in which the state μ_{t-1} is calculated from Ω_{t-1} and ξ_{t-1} in the obvious way. Line 5 computes the state $\bar{\mu}_t$ using the equation familiar from the EKF (Line 2 in Table 3.3).

The necessity to recover the state estimate seems at odds with the desire to represent the filter using its canonical parameters. We will revisit this topic when discussing the use of extended information filters in the context of robotic mapping.

3.4.5 Mathematical Derivation of the Extended Information Filter

The extended information filter is easily derived by essentially performing the same linearization that led to the extended Kalman filter above. As in (3.50) and (3.52), EIFs approximate g and h by a Taylor expansion:

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + G_t\,(x_{t-1} - \mu_{t-1}) && (3.88)\\ h(x_t) &\approx h(\bar{\mu}_t) + H_t\,(x_t - \bar{\mu}_t) && (3.89) \end{aligned}$$

Here G_t and H_t are the Jacobians of g and h at μ_{t-1} and $\bar{\mu}_t$, respectively:

$$\begin{aligned} G_t &= g'(u_t, \mu_{t-1}) && (3.90)\\ H_t &= h'(\bar{\mu}_t) && (3.91) \end{aligned}$$

These definitions are equivalent to those in the EKF. The prediction step is now derived from Lines 2 and 3 of the EKF algorithm (Table 3.3), which are restated here:

$$\begin{aligned} \bar{\Sigma}_t &= G_t\,\Sigma_{t-1}\,G_t^T + R_t && (3.92)\\ \bar{\mu}_t &= g(u_t, \mu_{t-1}) && (3.93) \end{aligned}$$

Substituting Σ_{t-1} by Ω_{t-1}^{-1} and $\bar{\mu}_t$ by $\bar{\Omega}_t^{-1}\bar{\xi}_t$ gives us the prediction equations of the extended information filter:

$$\begin{aligned} \bar{\Omega}_t &= (G_t\,\Omega_{t-1}^{-1}\,G_t^T + R_t)^{-1} && (3.94)\\ \bar{\xi}_t &= \bar{\Omega}_t\,g(u_t, \Omega_{t-1}^{-1}\,\xi_{t-1}) && (3.95) \end{aligned}$$

The measurement update is derived from Equations (3.59) and (3.60). In particular, (3.60) defines the following Gaussian posterior:

$$bel(x_t) = \eta\,\exp\big\{-\tfrac{1}{2}\,(z_t - h(\bar{\mu}_t) - H_t\,(x_t - \bar{\mu}_t))^T\,Q_t^{-1}$$

$$(z_t - h(\bar{\mu}_t) - H_t\ (x_t - \bar{\mu}_t)) - \tfrac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\Big\} \quad (3.96)$$

Multiplying out the exponent and reordering the terms gives us the following expression for the posterior:

$$\begin{aligned} bel(x_t) &= \eta\ \exp\big\{-\tfrac{1}{2}\ x_t^T\ H_t^T\ Q_t^{-1}\ H_t\ x_t + x_t^T\ H_t^T\ Q_t^{-1}\ [z_t - h(\bar{\mu}_t) - H_t\ \bar{\mu}_t] \\ &\quad -\tfrac{1}{2} x_t^T\ \bar{\Sigma}_t^{-1}\ x_t + x_t^T\ \bar{\Sigma}_t^{-1}\ \bar{\mu}_t\big\} \\ &= \eta\ \exp\big\{-\tfrac{1}{2}\ x_t^T\ \big[H_t^T\ Q_t^{-1}\ H_t + \bar{\Sigma}_t^{-1}\big]\ x_t \\ &\quad + x_t^T\ \big[H_t^T\ Q_t^{-1}\ [z_t - h(\bar{\mu}_t) - H_t\ \bar{\mu}_t] + \ \bar{\Sigma}_t^{-1}\ \bar{\mu}_t\big] \end{aligned} \quad (3.97)$$

With $\bar{\Sigma}_t^{-1} = \bar{\Omega}_t$ this expression resolves to the following information form:

$$\begin{aligned} bel(x_t) &= \eta\ \exp\big\{-\tfrac{1}{2}\ x_t^T\ \big[H_t^T\ Q_t^{-1}\ H_t + \bar{\Omega}_t\big]\ x_t \\ &\quad + x_t^T\ \big[H_t^T\ Q_t^{-1}\ [z_t - h(\bar{\mu}_t) - H_t\ \bar{\mu}_t] + \bar{\xi}_t\big] \end{aligned} \quad (3.98)$$

We can now read off the measurement update equations by collecting the terms in the squared brackets:

$$\Omega_t = \bar{\Omega}_t + H_t^T\ Q_t^{-1}\ H_t \quad (3.99)$$

$$\xi_t = \bar{\xi}_t + H_t^T\ Q_t^{-1}\ [z_t - h(\bar{\mu}_t) - H_t\ \bar{\mu}_t] \quad (3.100)$$

3.4.6 Practical Considerations

When applied to robotics problems, the information filter possesses several advantages over the Kalman filter. For example, representing global uncertainty is simple in the information filter: simply set $\Omega = 0$. When using moments, such global uncertainty amounts to a covariance of infinite magnitude. This is especially problematic when sensor measurements carry information about a strict subset of all state variables, a situation often encountered in robotics. Special provisions have to be made to handle such situations in EKFs. Furthermore, the information filter tends to be numerically more stable than the Kalman filter in many of the applications discussed later in this book.

Another advantage of the information filter over the Kalman filter arises from its natural fit for multi-robot problems. Multi-robot problems often involve the integration

of sensor data collected decentrally. Such integration is commonly performed through Bayes rule. When represented in logarithmic form, Bayes rule becomes an addition. As noted above, the canonical parameters of information filters represent a probability in logarithmic form. Thus, information integration is achieved by summing up information from multiple robots. Addition is commutative. Because of this, information filters can often integrate information in arbitrary order, with arbitrary delays, and in a completely decentralized manner. While the same is possible using the moments representation—after all, they represent the same information—the necessary overhead for doing so is much higher. Despite this advantage, the use of information filters in multi-robot systems remains largely under-explored.

These advantages of the information filter are offset by important limitations. A primary disadvantage of the EIF is the need to recover a state estimate in the update step, when applied to nonlinear systems. This step, if implemented as stated here, requires the inversion of the information matrix. Further matrix inversions are required for the prediction step of the information filters. In many robotics problems, the EKF does not involve the inversion of matrices of comparable size. For high dimensional state spaces, the information filter is generally believed to be computationally inferior to the Kalman filter. In fact, this is one of the reasons why the EKF has been vastly more popular than the EIF.

As we will see later in this book, these limitations do not necessarily apply to problems in which the information matrix possess structure. In many robotics problems, the interaction of state variables is local; as a result, the information matrix may be sparse. Such sparseness does *not* translate to sparseness of the covariance.

Information filters can be thought of as graphs, where states are connected whenever the corresponding off-diagonal element in the information matrix is non-zero. Sparse information matrices correspond to sparse graphs; in fact, such graphs are commonly known as Gaussian Markov random fields. A flurry of algorithms exist to perform the basic update and estimation equations efficiently for such fields, under names like "loopy belief propagation." In this book, we will encounter a mapping problem in which the information matrix is (approximately) sparse, and develop an extended information filter that is significantly more efficient than both Kalman filters and non-sparse information filters.

3.5 SUMMARY

In this section, we introduced efficient Bayes filter algorithms that represent the posterior by multivariate Gaussians. We noted that

- Gaussians can be represented in two different ways: The moments representation and the canonical representation. The moments representation consists of the mean (first moment) and the covariance (second moment) of the Gaussian. The canonical, or natural, representation consists of an information matrix and an information vector. Both representations are duals of each other, and each can be recovered from the other via matrix inversion.

- Bayes filters can be implemented for both representations. When using the moments representation, the resulting filter is called Kalman filter. The dual of the Kalman filter is the information filter, which represents the posterior in the canonical representation. Updating a Kalman filter based on a control is computationally simple, whereas incorporating a measurement is more difficult. The opposite is the case for the information filter, where incorporating a measurement is simple, but updating the filter based on a control is difficult.

- For both filters to calculate the correct posterior, three assumptions have to be fulfilled. First, the initial belief must be Gaussian. Second, the state transition probability must be composed of a function that is linear in its argument with added independent Gaussian noise. Third, the same applies to the measurement probability. It must also be linear in its argument, with added Gaussian noise. Systems that meet these assumptions are called linear Gaussian systems.

- Both filters can be extended to nonlinear problems. The technique described in this chapter calculates a tangent to the nonlinear function. Tangents are linear, making the filters applicable. The technique for finding a tangent is called Taylor expansion. Performing a Taylor expansion involves calculating the first derivative of the target function, and evaluating it at a specific point. The result of this operation is a matrix known as the Jacobian. The resulting filters are called "extended."

- The accuracy of Taylor series expansions depends on two factors: The degree of nonlinearity in the system, and the width of the posterior. Extended filters tend to yield good results if the state of the system is known with relatively high accuracy, so that the remaining covariance is small. The larger the uncertainty, the higher the error introduced by the linearization.

- One of the primary advantages of Gaussian filters is computational: The update requires time polynomial in the dimensionality of the state space. This is not

the case of some of the techniques described in the next chapter. The primary disadvantage is their confinement to unimodal Gaussian distributions.

- Within the multivariate Gaussian regime, both filters, the Kalman filter and the information filter, have orthogonal strengths and weaknesses. However, the Kalman filter and its nonlinear extension, the extended Kalman filter, are vastly more popular than the information filter.

The selection of the material in this chapter is based on today's most popular techniques in robotics. There exists a huge number of variations and extensions of the Gaussian filters presented here, that address the various limitations and shortcomings. One of the most apparent limitations of the material presented thus far is the fact that the posterior is represented by a single Gaussian. This confines these filters to situations where the posterior can be described by a unimodal distribution. This is often appropriate in tracking applications, where a robot tracks a state variable with limited uncertainty. When uncertainty grows more global, a single mode can be insufficient, and Gaussians become too crude an approximation to the true posterior belief. This limitation has been well recognized, and even within the Gaussian paradigm extensions exist that can represent multimodal beliefs, e.g., using mixtures of Gaussians. Popular non-Gaussian approaches are described in the next chapter.

3.6 BIBLIOGRAPHICAL REMARKS

Inversion lemma: G.H. Golub, C.F. Van Loan, Matrix Computations, North Oxford Academic, 1986.

4

NONPARAMETRIC FILTERS

A popular alternative to Gaussian techniques are nonparametric filters. Nonparametric filters do not rely on a fixed functional form of the posterior, such as Gaussians. Instead, they approximate posteriors by a finite number of values, each roughly corresponding to a region in state space. Some nonparametric Bayes filters rely on a decomposition of the state space, in which each such value corresponds to the cumulative probability of the posterior density in a compact subregion of the state space. Others approximate the state space by random samples drawn from the posterior distribution. In all cases, the number of parameters used to approximate the posterior can be varied. The quality of the approximation depends on the number of parameters used to represent the posterior. As the number of parameters goes to infinity, nonparametric techniques tend to converge uniformly to the correct posterior (under specific smoothness assumptions).

This chapter discusses two nonparametric approaches for approximating posteriors over continuous spaces with finitely many values. The first decomposes the state space into finitely many regions, and represents the posterior by a histogram. A histogram assigns to each region a single cumulative probability; they are best thought of as piecewise constant approximations to a continuous density. The second technique represents posteriors by finitely many samples. The resulting filter is known as particle filter and has gained immense popularity in certain robotics problems.

Both types of techniques, histograms and particle filters, do not make strong parametric assumptions on the posterior density. In particular, they are well-suited to represent complex multimodal beliefs. For this reason, they are often the method of choice when a robot has to cope with phases of global uncertainty, and when it faces hard data association problems that yield separate, distinct hypotheses. However, the representational power of these techniques comes at the price of added computational com-

1: **Algorithm Discrete_Bayes_filter(**$\{p_{k,t-1}\}, u_t, z_t$**):**
2: for all k do
3: $\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\ p_{i,t-1}$
4: $p_{k,t} = \eta\ p(z_t \mid X_t = x_k)\ \bar{p}_{k,t}$
5: endfor
6: return $\{p_{k,t}\}$

Table 4.1 The discrete Bayes filter. Here x_i, x_k denote individual states.

plexity. Fortunately, both nonparametric techniques described in this chapter make it possible to adapt the number of parameters to the (suspected) complexity of the posterior. When the posterior is of low complexity (e.g., focused on a single state with a small margin of uncertainty), they use only small numbers of parameters. For complex posteriors, e.g., posteriors with many modes scattered across the state space, the number of parameters grows larger.

Techniques that can adapt the number of parameters to represent the posterior online are called *adaptive*. They are called *resource-adaptive* if they can adapt based on the computational resources available for belief computation. Resource-adaptive techniques play an important role in robotics. They enable robots to make decisions in real time, regardless of the computational resources available. Particle filters are often implemented as a resource-adaptive algorithm, by adapting the number of particles online based on the available computational resources.

4.1 THE HISTOGRAM FILTER

Histogram filters decompose the state space into finitely many regions, and represent the cumulative posterior for each region by a single probability value. When applied to discrete spaces, such filters are known as *discrete Bayes filters*. In continuous state spaces, they are known as *histogram filters*. We will first describe the discrete Bayes filter, and then discuss its use in continuous state spaces.

4.1.1 The Discrete Bayes Filter Algorithm

Discrete Bayes filters apply to problems with *finite* state spaces, that is, where the random variable X_t can take on finitely many values. We already encountered a discrete Bayes filter in Section 2.4.2, when discussing the example of a robot estimating the probability that a door is open. Some of the robotic mapping problems discussed in later chapters also involve discrete random variables. For example, occupancy grid mapping algorithms assume that each location in the environment is either occupied or free. The corresponding random variable is binary, that is, it can take on two different values. Thus, finite state spaces play an important role in robotics.

Table 4.1 provides pseudo-code for the discrete Bayes filter. This code is derived from the general Bayes filter in Table 2.1 by replacing the integration with a finite sum. The variables x_i and x_k denote individual states, of which there may only be finitely many. The belief at time t is an assignment of a probability to each state x_k, denoted $p_{k,t}$. Thus, the input to the algorithm is a discrete probability distribution $\{p_{k,t}\}$, along with the most recent control u_t and measurement z_t. Line 3 calculates the prediction, the belief for the new state based on the control alone. This prediction is then updated in Line 4, so as to incorporate the measurement. The discrete Bayes filter algorithm is popular in many areas of signal processing (such as speech recognition), where it is often referred to as the forward pass of a hidden Markov models.

4.1.2 Continuous State

Of particular interest in this book will be the use of discrete Bayes filters as an approximate inference tool for *continuous* state spaces. Such filters are called *histogram filters*. Histogram filters decompose a continuous state space into finitely many regions:

$$\text{range}(X_t) \quad = \quad \mathbf{x}_{1,t} \cup \mathbf{x}_{2,t} \cup \ldots \mathbf{x}_{K,t} \tag{4.1}$$

Here X_t is the familiar random variable describing the state of the robot at time t. The function $\text{range}(X_t)$ denotes the state space, that is, the universe of possible values that X_t might assume. Each $\mathbf{x}_{k,t}$ describes a convex region. These regions together form a partitioning of the state space, that is, for each $i \neq k$ we have $\mathbf{x}_{i,t} \cap \mathbf{x}_{k,t} = \emptyset$ and $\bigcup_k \mathbf{x}_{k,t} = \text{range}(X_t)$. A straightforward decomposition of a continuous state space is a multi-dimensional grid, where each $\mathbf{x}_{k,t}$ is a grid cell. Through the granularity of the decomposition, we can trade off accuracy and computational efficiency. Fine-

grained decompositions infer smaller approximation errors than coarse ones, but at the expense of increased computational complexity.

As we already discussed, the discrete Bayes filter assigns to each region $\mathbf{x}_{k,t}$ a probability, $p_{k,t}$. Within each region, the discrete Bayes filter carries no further information on the belief distribution. Thus, the posterior becomes a piecewise constant PDF, which assigns a uniform probability to each state x_t within each region $\mathbf{x}_{k,t}$:

$$p(x_t) \quad = \quad \frac{p_{k,t}}{|\mathbf{x}_{k,t}|} \tag{4.2}$$

Here $|\mathbf{x}_{k,t}|$ is the volume of the region $\mathbf{x}_{k,t}$.

If the state space is truly discrete, the conditional probabilities $p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1})$ and $p(z_t \mid \mathbf{x}_{k,t})$ are well-defined, and the algorithm can be implemented as stated. In continuous state spaces, one is usually given the densities $p(x_t \mid u_t, x_{t-1})$ and $p(z_t \mid x_t)$, which are defined for individual states (and not for regions in state space). For cases where each region $\mathbf{x}_{k,t}$ is small and of the same size, these densities are usually approximated by substituting $\mathbf{x}_{k,t}$ by a representative of this region. For example, we might simply "probe" using the mean state in $\mathbf{x}_{k,t}$

$$\hat{x}_{k,t} \quad = \quad |\mathbf{x}_{k,t}|^{-1} \int_{\mathbf{x}_{k,t}} x_t \; dx_t \tag{4.3}$$

One then simply replaces

$$p(z_t \mid \mathbf{x}_{k,t}) \quad \approx \quad p(z_t \mid \hat{x}_{k,t}) \tag{4.4}$$

$$p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \quad \approx \quad \frac{\eta}{|\mathbf{x}_{k,t}|} \, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1}) \tag{4.5}$$

These approximations are the result of the piecewise uniform interpretation of the discrete Bayes filter stated in (4.2), and a linearization similar to the one used by EKFs.

To see that (4.4) is a reasonable approximation, we note that $p(z_t \mid \mathbf{x}_{k,t})$ can be expressed as the following integral:

$$p(z_t \mid \mathbf{x}_{k,t}) \quad = \quad \frac{p(z_t, \mathbf{x}_{k,t})}{p(\mathbf{x}_{k,t})}$$

$$
\begin{aligned}
&= \frac{\int_{\mathbf{x}_{k,t}} p(z_t, x_t)\, dx_t}{\int_{\mathbf{x}_{k,t}} p(x_t)\, dx_t} \\
&= \frac{\int_{\mathbf{x}_{k,t}} p(z_t \mid x_t)\, p(x_t)\, dx_t}{\int_{\mathbf{x}_{k,t}} p(x_t)\, dx_t} \\
&\overset{(4.2)}{=} \frac{\int_{\mathbf{x}_{k,t}} p(z_t \mid x_t)\, \frac{p_{k,t}}{|\mathbf{x}_{k,t}|}\, dx_t}{\int_{\mathbf{x}_{k,t}} \frac{p_{k,t}}{|\mathbf{x}_{k,t}|}\, dx_t} \\
&= \frac{\int_{\mathbf{x}_{k,t}} p(z_t \mid x_t)\, dx_t}{\int_{\mathbf{x}_{k,t}} 1\, dx_t} \\
&= |\mathbf{x}_{k,t}|^{-1} \int_{\mathbf{x}_{k,t}} p(z_t \mid x_t)\, dx_t
\end{aligned} \tag{4.6}
$$

This expression is an exact description of the desired probability under the piecewise uniform distribution model in (4.2). If we now approximate $p(z_t \mid x_t)$ by $p(z_t \mid \hat{x}_{k,t})$ for $x_t \in \mathbf{x}_{k,t}$, we obtain

$$
\begin{aligned}
p(z_t \mid \mathbf{x}_{k,t}) &\approx |\mathbf{x}_{k,t}|^{-1} \int_{\mathbf{x}_{k,t}} p(z_t \mid \hat{x}_{k,t})\, dx_t \\
&= |\mathbf{x}_{k,t}|^{-1}\, p(z_t \mid \hat{x}_{k,t}) \int_{\mathbf{x}_{k,t}} 1\, dx_t \\
&= |\mathbf{x}_{k,t}|^{-1}\, p(z_t \mid \hat{x}_{k,t})\, |\mathbf{x}_{k,t}| \\
&= p(z_t \mid \hat{x}_{k,t})
\end{aligned} \tag{4.7}
$$

which is the approximation stated above in (4.4).

The derivation of the approximation to $p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1})$ in (4.5) is slightly more involved, since regions occur on both sides of the conditioning bar. In analogy to our transformation above, we obtain:

$$
\begin{aligned}
&p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \\
&\quad = \frac{p(\mathbf{x}_{k,t}, \mathbf{x}_{i,t-1} \mid u_t)}{p(\mathbf{x}_{i,t-1} \mid u_t)} \\
&\quad = \frac{\int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t, x_{t-1} \mid u_t)\, dx_t, dx_{t-1}}{\int_{\mathbf{x}_{i,t-1}} p(x_{t-1} \mid u_t)\, dx_{t-1}}
\end{aligned}
$$

$$= \frac{\int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t \mid u_t, x_{t-1})\, p(x_{t-1} \mid u_t)\, dx_t, dx_{t-1}}{\int_{\mathbf{x}_{i,t-1}} p(x_{t-1} \mid u_t)\, dx_{t-1}} \tag{4.8}$$

We now exploit the Markov assumption, which implies independence between x_{t-1} and u_t, and thus $p(x_{t-1} \mid u_t) = p(x_{t-1})$:

$$\begin{aligned}
& p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \\
&= \frac{\int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t \mid u_t, x_{t-1})\, p(x_{t-1})\, dx_t, dx_{t-1}}{\int_{\mathbf{x}_{i,t-1}} p(x_{t-1})\, dx_{t-1}} \\
&= \frac{\int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t \mid u_t, x_{t-1})\, \frac{p_{i,t-1}}{|\mathbf{x}_{i,t-1}|}\, dx_t, dx_{t-1}}{\int_{\mathbf{x}_{i,t-1}} \frac{p_{i,t-1}}{|\mathbf{x}_{i,t-1}|}\, dx_{t-1}} \\
&= \frac{\int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t \mid u_t, x_{t-1})\, dx_t, dx_{t-1}}{\int_{\mathbf{x}_{i,t-1}} 1\, dx_{t-1}} \\
&= |\mathbf{x}_{i,t-1}|^{-1} \int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(x_t \mid u_t, x_{t-1})\, dx_t, dx_{t-1}
\end{aligned} \tag{4.9}$$

If we now approximate $p(x_t \mid u_t, x_{t-1})$ by $p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})$ as before, we obtain the following approximation. Note that the normalizer η becomes necessary to ensure that the approximation is a valid probability distribution:

$$\begin{aligned}
& p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \\
&\approx \eta\, |\mathbf{x}_{i,t-1}|^{-1} \int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})\, dx_t, dx_{t-1} \\
&= \eta\, |\mathbf{x}_{i,t-1}|^{-1}\, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1}) \int_{\mathbf{x}_{k,t}} \int_{\mathbf{x}_{i,t-1}} 1\, dx_t, dx_{t-1} \\
&= \eta\, |\mathbf{x}_{i,t-1}|^{-1}\, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1}) |\mathbf{x}_{k,t}|\, |\mathbf{x}_{i,t-1}| \\
&= \eta\, |\mathbf{x}_{k,t}|\, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})
\end{aligned} \tag{4.10}$$

If all regions are of equal size (meaning that $|\mathbf{x}_{k,t}|$ is the same for all k), we can simply omit the factor $|\mathbf{x}_{k,t}|$, since it is subsumed by the normalizer. The resulting discrete Bayes filter is then equivalent to the algorithm outlined in Table 4.1. If implemented as stated there, the auxiliary parameters $\bar{p}_k$ do not constitute a probability distribution, since they are not normalized (compare Line 3 to (4.10)). However, normalization takes place in Line 4, so that the output parameters are indeed a valid probability distribution.

4.1.3 Decomposition Techniques

In robotics, decomposition techniques of continuous state spaces come into two basic flavors: *static* and *dynamic*. Static techniques rely on a fixed decomposition that is chosen in advance, irrespective of the shape of the posterior that is being approximated. Dynamic techniques adapt the decomposition to the specific shape of the posterior distribution. Static techniques are usually easier to implement, but they can be wasteful with regards to computational resources.

A primary example of a dynamic decomposition technique is the family of *density trees*. Density trees decompose the state space recursively, in ways that adapt the resolution to the posterior probability mass. The intuition behind this decomposition is that the level of detail in the decomposition is a function of the posterior probability: The less likely a region, the coarser the decomposition. Figure 4.1 illustrates the difference between a static grid representation and a density tree representation for a two-dimensional probability density. While both representations have the same approximation quality, the density tree is more compact than the grid representation. Dynamic techniques like density trees can often cut the computation complexity by orders of magnitude over static ones, yet they require additional implementation effort.

An effect similar to that of dynamic decompositions can be achieved by *selective updating*. When updating a posterior represented by a grid, selective techniques update a fraction of all grid cells only. A common implementation of this idea updates only those grid cells whose posterior probability exceeds a user-specified threshold. Selective updating can be viewed as a hybrid decomposition, which decomposes the state space into a fine grained grid and one large set that contains all regions not chosen by the selective update procedure. In this light, it can be thought of as a dynamic decomposition technique, since the decision as to which grid cells to consider during the update is made online, based on the shape of the posterior distribution. Selective updating techniques can reduce the computational effort involved in updating beliefs by many orders of magnitude. They make it possible to use grid decompositions in spaces of three or more dimensions.

The mobile robotics literature often distinguishes topological from metric representations of space. While no clear definition of these terms exist, topological representations are often thought of as course graph-like representations, where nodes in the graph correspond to significant places (or features) in the environment. For indoor environments, such places may correspond to intersections, T-junctions, dead ends, and so on. The resolution of such decompositions, thus, depends on the structure of the environment. Alternatively, one might decompose the state space using regularly-

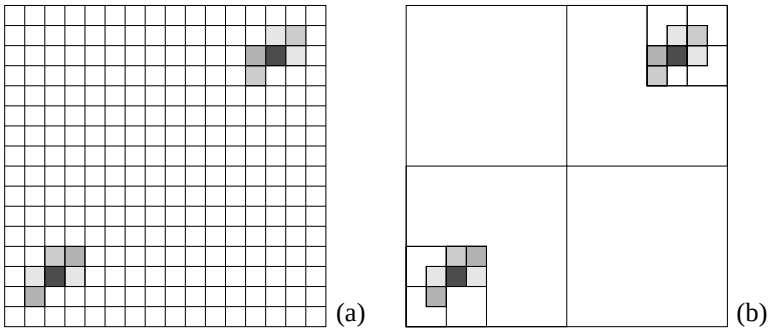

Figure 4.1 (a) Grid based representation of a two-dimensional probability density. The probability density concentrates on the upper right and lower left part of the state space. (b) Density tree representation of the same distribution.

spaced grids. Such a decomposition does not depend on the shape and location of the environmental features. Grid representations are often thought of as metric although, strictly speaking, it is the embedding space that is metric, not the decomposition. In mobile robotics, the spatial resolution of grid representations tends to be higher than that of topological representations. For instance, some of the examples in Chapter 7 use grid decompositions with cell sizes of 10 centimeters or less. This increased accuracy comes at the expense of increased computational costs.

4.1.4 Binary Bayes Filters With Static State

Certain problems in robotics are best formulated as estimation problems with binary state that does not change over time. Problems of this type arise if a robot estimates a fixed binary quantity in the environment from a sequence of sensor measurements. Since the state is static, the belief is a function of the measurements:

$$bel_t(x) \quad = \quad p(x \mid z_{1:t}, u_{1:t}) \quad = \quad p(x \mid z_{1:t}) \tag{4.11}$$

where the state is chosen from two possible values, denoted by x and $\neg x$. In particular, we have $bel_t(\neg x) = 1 - bel_t(x)$. The lack of a time index for the state x reflects the fact that the state does not change.

Naturally, binary estimation problems of this type can be tackled using the discrete Bayes filter in Table 4.1. However, the belief is commonly implemented as a *log odds ratio.* The *odds* of a state x is defined as the ratio of the probability of this event

1: **Algorithm binary_Bayes_filter(l_{t-1}, z_t):**
2: $l_t = l_{t-1} + \log \frac{p(x|z_t)}{1-p(x|z_t)} - \log \frac{p(x)}{1-p(x)}$
3: return l_t

Table 4.2 The binary Bayes filter in log odds form with an inverse measurement model. Here l_t is the log odds of the posterior belief over a binary state variable that does not change over time.

divided by the probability of its negate

$$\frac{p(x)}{p(\neg x)} \quad = \quad \frac{p(x)}{1 - p(x)} \tag{4.12}$$

The log odds is the logarithm of this expression

$$l(x) \quad := \quad \log \frac{p(x)}{1 - p(x)} \, . \tag{4.13}$$

Log odds assume values from $-\infty$ to ∞. The Bayes filter for updating beliefs in log odds representation is computationally elegant. It avoids truncation problems that arise for probabilities close to 0 or 1.

Table 4.2 provides the basic update algorithm. This algorithm is additive; in fact, any algorithm that increment and decrement a variable in response to measurements can be interpreted as a Bayes filter in log odds form. This binary Bayes filter uses an *inverse measurement model* $p(x \mid z_t)$, instead of the familiar forward model $p(z_t \mid x)$. The inverse measurement model specifies a distribution over the (binary) state variable as a function of the measurement z_t. Inverse models are often used in situations where measurements are more complex than the binary state. An example of such a situation is the problem of estimating whether or not a door is closed, from camera images. Here the state is extremely simple, but the space of all measurements is huge. It is easier to devise a function that calculates a probability of a door being closed from a camera image, than describing the distribution over all camera images that show a closed door. In other words, it is easier to implement an inverse than a forward sensor model.

As the reader easily verifies from our definition of the log odds (4.13), the belief $bel_t(x)$ can be recovered from the log odds ratio l_t by the following equation:

$$bel_t(x) \quad = \quad 1 - \frac{1}{1 + \exp\{l_t\}} \tag{4.14}$$

To verify the correctness of our binary Bayes filter algorithm, we briefly restate the basic filter equation with the Bayes normalizer made explicit:

$$\begin{aligned} p(x \mid z_{1:t}) &= \frac{p(z_t \mid x, z_{1:t-1})\, p(x \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})} \\ &= \frac{p(z_t \mid x)\, p(x \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})} \end{aligned} \tag{4.15}$$

We now apply Bayes rule to the measurement model $p(z_t \mid x)$:

$$p(z_t \mid x) \quad = \quad \frac{p(x \mid z_t)\, p(z_t)}{p(x)} \tag{4.16}$$

and obtain

$$p(x \mid z_{1:t}) \quad = \quad \frac{p(x \mid z_t)\, p(z_t)\, p(x \mid z_{1:t-1})}{p(x)\, p(z_t \mid z_{1:t-1})}. \tag{4.17}$$

By analogy, we have for the opposite event $\neg x$:

$$p(\neg x \mid z_{1:t}) \quad = \quad \frac{p(\neg x \mid z_t)\, p(z_t)\, p(\neg x \mid z_{1:t-1})}{p(\neg x)\, p(z_t \mid z_{1:t-1})} \tag{4.18}$$

Dividing (4.17) by (4.18) leads to cancellation of various difficult-to-calculate probabilities:

$$\begin{aligned} \frac{p(x \mid z_{1:t})}{p(\neg x \mid z_{1:t})} &= \frac{p(x \mid z_t)}{p(\neg x \mid z_t)} \, \frac{p(x \mid z_{1:t-1})}{p(\neg x \mid z_{1:t-1})} \, \frac{p(\neg x)}{p(x)} \\ &= \frac{p(x \mid z_t)}{1 - p(x \mid z_t)} \, \frac{p(x \mid z_{1:t-1})}{1 - p(x \mid z_{1:t-1})} \, \frac{1 - p(x)}{p(x)} \end{aligned} \tag{4.19}$$

We denote the log odds ratio of the belief $bel_t(x)$ by $l_t(x)$. The log odds belief at time t is given by the logarithm of (4.19).

$$\begin{aligned} l_t(x) &= \log \frac{p(x \mid z_t)}{1 - p(x \mid z_t)} + \log \frac{p(x \mid z_{1:t-1})}{1 - p(x \mid z_{1:t-1})} + \log \frac{1 - p(x)}{p(x)} \\ &= \log \frac{p(x \mid z_t)}{1 - p(x \mid z_t)} - \log \frac{p(x)}{1 - p(x)} + l_{t-1}(x) \end{aligned} \tag{4.20}$$

Here $p(x)$ is the *prior* probability of the state x. As (4.20), each measurement update involves the addition of the prior (in log odds form). The prior also defines the log odds of the initial belief before processing any sensor measurement:

$$l_0(x) = \log \frac{1 - p(x)}{p(x)} \tag{4.21}$$

4.2 THE PARTICLE FILTER

4.2.1 Basic Algorithm

The particle filter is an alternative nonparametric implementation of the Bayes filter. Just like histogram filters, particle filters approximate the posterior by a finite number of parameters. However, they differ in the way these parameters are generated, and in which they populate the state space. The key idea of the particle filter is to represent the posterior $bel(x_t)$ by a set of random state samples drawn from this posterior. Figure **??** illustrates this idea for a Gaussian. Instead of representing the distribution by a parametric form (the exponential function that defines the density of a normal distribution), particle filters represent a distribution by a set of samples drawn from this distribution. Such a representation is approximate, but it is nonparametric, and therefore can represent a much broader space of distributions than, for example, Gaussians.

In particle filters, the samples of a posterior distribution are called *particles* and are denoted

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \tag{4.22}$$

Each particle $x_t^{[m]}$ (with $1 \leq m \leq M$) is a concrete instantiation of the state at time t, that is, a hypothesis as to what the true world state may be at time t. Here M denotes

1: **Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Table 4.3 The particle filter algorithm, a variant of the Bayes filter based on importance sampling.

the number of particles in the particle set $\mathcal{X}_t$. In practice, the number of particles M is often a large number, e.g., $M = 1,000$. In some implementations M is a function of t or of other quantities related to the belief $bel(x_t)$.

The intuition behind particle filters is to approximate the belief $bel(x_t)$ by the set of particles $\mathcal{X}_t$. Ideally, the likelihood for a state hypothesis x_t to be included in the particle set $\mathcal{X}_t$ shall be proportional to its Bayes filter posterior $bel(x_t)$:

$$x_t^{[m]} \quad \sim \quad p(x_t \mid z_{1:t}, u_{1:t}) \tag{4.23}$$

As a consequence of (4.23), the denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region. As we will discuss below, the property (4.23) holds only asymptotically for $M \uparrow \infty$ for the standard particle filter algorithm. For finite M, particles are drawn from a slightly different distribution. In practice, this difference is negligible as long as the number of particles is not too small (e.g., $M \geq 100$).

Just like all other Bayes filter algorithms discussed thus far, the particle filter algorithm constructs the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$ one time step earlier. Since beliefs are represented by sets of particles, this means that particle filters

construct the particle set $\mathcal{X}_t$ recursively from the set $\mathcal{X}_{t-1}$. The most basic variant of the particle filter algorithm is stated in Table 4.3. The input of this algorithm is the particle set $\mathcal{X}_{t-1}$, along with the most recent control u_t and the most recent measurement z_t. The algorithm then first constructs a temporary particle set $\bar{\mathcal{X}}$ which is reminiscent (but not equivalent) to the belief $\overline{bel}(x_t)$. It does this by systematically processing each particle $x_{t-1}^{[m]}$ in the input particle set $\mathcal{X}_{t-1}$ as follows.

1. Line 4 generates a hypothetical state $x_t^{[m]}$ for time t based on the particle $x_{t-1}^{[m]}$ and the control u_t. The resulting sample is indexed by m, indicating that it is generated from the m-th particle in $\mathcal{X}_{t-1}$. This step involves sampling from the next state distribution $p(x_t \mid u_t, x_{t-1})$. To implement this step, one needs to be able to sample from $p(x_t \mid u_t, x_{t-1})$. The ability to sample from the state transition probability is not given for arbitrary distributions $p(x_t \mid u_t, x_{t-1})$. However, many major distributions in this book possess efficient algorithms for generating samples. The set of particles resulting from iterating Step 4 M times is the filter's representation of $\overline{bel}(x_t)$.

2. Line 5 calculates for each particle $x_t^{[m]}$ the so-called *importance factor*, denoted $w_t^{[m]}$. Importance factors are used to incorporate the measurement z_t into the particle set. The importance, thus, is the probability of the measurement z_t under the particle $x_t^{[m]}$, that is, $w_t^{[m]} = p(z_t \mid x_t^{[m]})$. If we interpret $w_t^{[m]}$ as the *weight* of a particle, the set of weighted particles represents (in approximation) the Bayes filter posterior $bel(x_t)$.

3. The real "trick" of the particle filter algorithm occurs in Lines 8 through 11 in Table 4.3. These lines implemented what is known as *resampling* or *importance resampling*. The algorithm draws with replacement M particles from the temporary set $\bar{\mathcal{X}}_t$. The probability of drawing each particle is given by its importance weight. Resampling transforms a particle set of M particles into another particle set of the same size. By incorporating the importance weights into the resampling process, the distribution of the particles change: whereas before the resampling step, they were distribution according to $\overline{bel}(x_t)$, after the resampling they are distributed (approximately) according to the posterior $bel(x_t) = \eta\ p(z_t \mid x_t^{[m]})\overline{bel}(x_t)$. In fact, the resulting sample set usually possesses many duplicates, since particles are drawn with replacement. More important are the particles that are *not* contained in $\mathcal{X}_t$: those tend to be the particles with lower importance weights.

The resampling step has the important function to force particles back to the posterior $bel(x_t)$. In fact, an alternative (and usually inferior) version of the particle filter would never resample, but instead would maintain for each particle an importance weight

that is initialized by 1 and updated multiplicatively:

$$w_t^{[m]} \quad = \quad p(z_t \mid x_t^{[m]})\, w_{t-1}^{[m]} \tag{4.24}$$

Such a particle filter algorithm would still approximate the posterior, but many of its particles would end up in regions of low posterior probability. As a result, it would require many more particles; how many depends on the shape of the posterior. The resampling step is a probabilistic implementation of the Darwinian idea of *survival of the fittest*: It refocuses the particle set to regions in state space with high posterior probability. By doing so, it focuses the computational resources of the filter algorithm to regions in the state space where they matter the most.

4.2.2 Importance Sampling

For the derivation of the particle filter, it shall prove useful to discuss the resampling step in more detail. Figure 4.2 illustrates the intuition behind the resampling step. Figure 4.2a shows a density function f of a probability distribution called the *target distribution*. What we would like to achieve is to obtain a sample from f. However, sampling from f directly may not be possible. Instead, we can generate particles from a related density, labeled g in Figure 4.2b. The distribution that corresponds to the density g is called *proposal distribution*. The density g must be such that $f(x) > 0$ implies $g(x) > 0$, so that there is a non-zero probability to generate a particle when sampling from g for any state that might be generated by sampling from f. However, the resulting particle set, shown at the bottom of Figure 4.2b, is distributed according to g, not to f. In particular, for any interval $A \subseteq \mathrm{range}(X)$ (or more generally, any Borel set A) the empirical count of particles that fall into A converges to the integral of g under A:

$$\frac{1}{M} \sum_{m=1}^{M} I(x^{[m]} \in A) \quad \longrightarrow \quad \int_A g(x)\, dx \tag{4.25}$$

To offset this difference between f and g, particles $x^{[m]}$ are weighted by the quotient

$$w^{[m]} \quad = \quad \frac{f(x^{[m]})}{g(x^{[m]})} \tag{4.26}$$

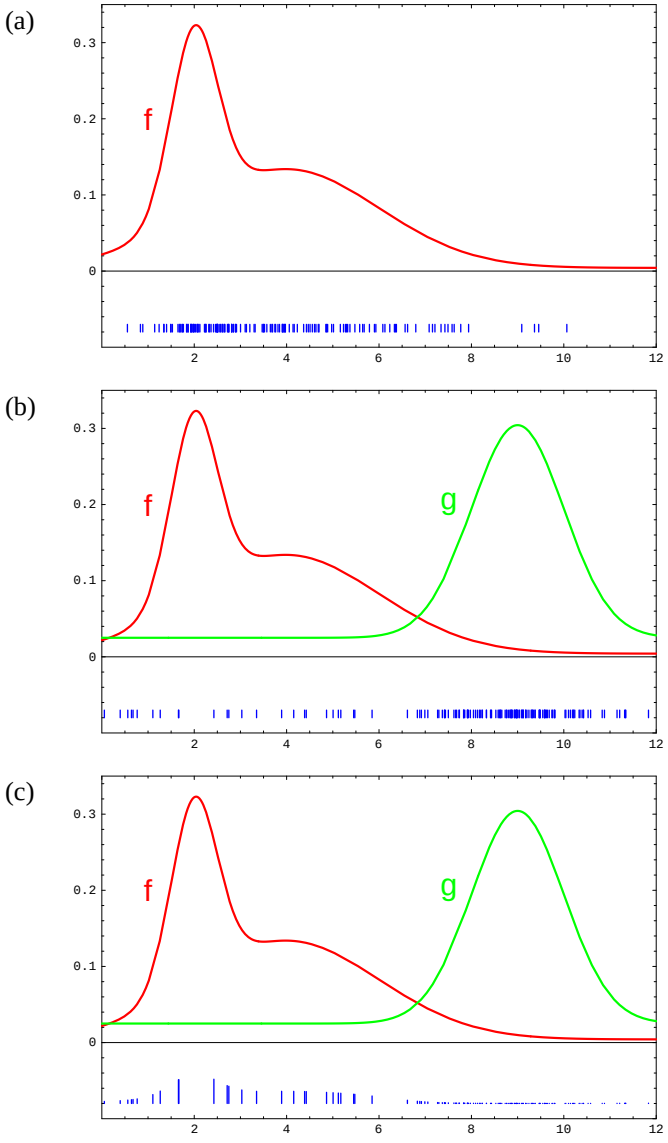

Figure 4.2 Illustration of importance factors in particle filters: (a) We seek to approximate the target density f. (b) Instead of sampling from f directly, we can only generate samples from a different density, g. Samples drawn from g are shown at the bottom of this diagram. (c) A sample of f is obtained by attaching the weight $f(x)/g(x)$ to each sample x. In particle filters, f corresponds to the belief $bel(x_t)$ and g to the belief $\overline{bel}(x_t)$.

This is illustrated by Figure 4.2c: The vertical bars in this figure indicate the magnitude of the importance weights. Importance weights are the non-normalized probability mass of each particle. In particular, we have

$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} I(x^{[m]} \in A)\, w^{[m]} \quad \longrightarrow \quad \int_A f(x)\, dx \tag{4.27}$$

where the first term serves as the normalizer for all importance weights. In other words, even though we generated the particles from the density g, the appropriately weighted particles converge to the density f.

The specific convergence involves an integration over a set A. Clearly, a particle set represents a discrete distribution, whereas f is continuous in our example. Because of this, there is no density that could be associated with a set of particles. The convergence, thus, is over the cumulative distribution function of f, not the density itself (hence the integration over A). A nice property of importance sampling is that it converges to the true density if $g(x) > 0$ whenever $f(x) > 0$. In most cases, the rate of convergence is in $O(\frac{1}{\sqrt{M}})$, where M is the number of samples. The constant factor depends on the similarity of $f(s)$ and $g(s)$.

In particle filters, the density f corresponds to the target belief $bel(x_t)$. Under the (asymptotically correct) assumption that the particles in $\mathcal{X}_{t-1}$ are distributed according to $bel(x_{t-1})$, the density g corresponds to the product distribution:

$$p(x_t \mid u_t, x_{t-1})\; bel(x_{t-1}) \tag{4.28}$$

This distribution is called the *proposal distribution*.

4.2.3 Mathematical Derivation of the PF

To derive particle filters mathematically, it shall prove useful to think of particles as samples of state sequences

$$x_{0:t}^{[m]} \quad = \quad x_0^{[m]}, x_1^{[m]}, \ldots, x_t^{[m]} \tag{4.29}$$

It is easy to modify the algorithm accordingly: Simply append to the particle $x_t^{[m]}$ the sequence of state samples from which it was generated $x_{0:t-1}^{[m]}$. This particle filter

calculates the posterior over all state sequences:

$$bel(x_{0:t}) \quad = \quad p(x_{0:t} \mid u_{1:t}, z_{1:t}) \tag{4.30}$$

instead of the belief $bel(x_t) = p(x_t \mid u_{1:t}, z_{1:t})$. Admittedly, the space over all state sequences is huge, and covering it with particles is usually plainly infeasible. However, this shall not deter us here, as this definition serves only as the means to derive the particle filter algorithm in Table 4.2.

The posterior $bel(x_{0:t})$ is obtained analogously to the derivation of $bel(x_t)$ in Section 2.4.3. In particular, we have

$$\begin{aligned}
& p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
& \overset{\text{Bayes}}{=} \quad \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& = \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned} \tag{4.31}$$

Notice the absence of integral signs in this derivation, which is the result of maintaining all states in the posterior, not just the most recent one as in Section 2.4.3.

The derivation is now carried out by induction. The initial condition is trivial to verify, assuming that our first particle set is obtained by sampling the prior $p(x_0)$. Let us assume that the particle set at time $t-1$ is distributed according to $bel(x_{0:t-1})$. For the m-th particle $x_{0:t-1}^{[m]}$ in this set, the sample $x_t^{[m]}$ generated in Step 4 of our algorithm is generated from the proposal distribution:

$$\begin{aligned}
& p(x_t \mid x_{t-1}, u_t)\, bel(x_{0:t-1}) \\
& \quad = \quad p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{0:t-1}, u_{0:t-1})
\end{aligned} \tag{4.32}$$

With

$$\begin{aligned}
w_t^{[m]} \quad &= \quad \frac{\text{target distribution}}{\text{proposal distribution}} \\
&= \quad \frac{\eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}{p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{0:t-1}, u_{0:t-1})} \\
&= \quad \eta\, p(z_t \mid x_t)
\end{aligned} \tag{4.33}$$

The constant η plays no role since the resampling takes place with probabilities *proportional* to the importance weights. By resampling particles with probability proportional to $w_t^{[m]}$, the resulting particles are indeed distributed according to the product of the proposal and the importance weights $w_t^{[m]}$:

$$\eta\, w_t^{[m]}\; p(x_t \mid x_{t-1}, u_t)\; p(x_{0:t-1} \mid z_{0:t-1}, u_{0:t-1}) \quad = \quad bel(x_{0:t}) \tag{4.34}$$

(Notice that the constant factor η here differs from the one in (4.33).) The algorithm in Table 4.2 follows now from the simple observation that if $x_{0:t}^{[m]}$ is distributed according to $bel(x_{0:t})$, then the state sample $x_t^{[m]}$ is (trivially) distributed according to $bel(x_t)$.

As we will argue below, this derivation is only correct for $M \longrightarrow \infty$, due to a laxness in our consideration of the normalization constants. However, even for finite M it explains the intuition behind the particle filter.

4.2.4 Properties of the Particle Filter

Particle filters are approximate and as such subject to approximation errors. There are four complimentary sources of approximation error, each of which gives rise to improved versions of the particle filter.

1. The first approximation error relates to the fact that only finitely many particles are used. This artifact introduces a systematic *bias* in the posterior estimate. To see, consider the extreme case of $M = 1$ particle. In this case, the loop in Lines 3 through 7 in Table 4.3 will only be executed once, and $\bar{\mathcal{X}}_t$ will contain only a single particle, sampled from the motion model. The key insight is that the resampling step (Lines 8 through 11 in Table 4.3) will now *deterministically* accept this sample, regardless of its importance factor $w_t^{[m]}$. Put differently, the measurement probability $p(z_t \mid x_t^{[m]})$ plays no role in the result of the update, and neither does z_t. Thus, if $M = 1$, the particle filter generates particles from the probability

   $$p(x_t \mid u_{1:t}) \tag{4.35}$$

   instead of the desired posterior $p(x_t \mid u_{1:t}, z_{1:t})$. It flatly ignores all measurements. How can this happen?

   The culprit is the normalization, implicit in the resampling step. When sampling in proportion to the importance weights (Line 9 of the algorithm), $w_t^{[m]}$ becomes

its own normalizer if $M = 1$:

$$p(\text{draw } x_t^{[m]} \text{ in Line 9}) \quad = \quad \frac{w_t^{[m]}}{w_t^{[m]}} \quad = 1 \tag{4.36}$$

In general, the problem is that the non-normalized values $w_t[m]$ are drawn from an M-dimensional space, but after normalization they reside in a space of dimension $M-1$. This is because after normalization, the m-th weight can be recovered from the $M-1$ other weights by subtracting those from 1. Fortunately, for larger values of M, the effect of loss of dimensionality, or degrees of freedom, becomes less and less pronounced.

2. A second source of error in the particle filter relates to the randomness introduced in the resampling phase. To understand this error, it will once again be useful to consider the extreme case, which is that of a robot whose state does not change. Sometimes, we know for a fact that $x_t = x_{t-1}$. A good example is that of mobile robot localization, for a non-moving robot. Let us furthermore assume that the robot possesses no sensors, hence it cannot estimate the state, and that it is unaware of the state. Initially, our particle set $\mathcal{X}_0$ will be generated from the prior; hence particles will be spread throughout the state space. The random nature of the resampling step (Line 8 in the algorithm) will regularly fail to draw a state sample $x^{[m]}$. However, since our state transition is deterministic, no new states will be introduced in the forward sampling step (Line 4). The result is quite daunting: With probability one, M identical copies of a single state will survive; the diversity will disappear due to the repetitive resampling. To an outside observer, it may appear that the robot has uniquely determined the world state—an apparent contradiction to the fact that the robot possesses no sensors.

   This example hints at an important limitation of particle filters with immense practical ramifications. In particular, the resampling process induces a loss of diversity in the particle population, which in fact manifests itself as approximation error. Such error is called *variance* of the estimator: Even though the variance of the particle set itself decreases, the variance of the particle set as an estimator of the true belief increases. Controlling this variance, or error, of the particle filter is essential for any practical implementation.

   There exist two major strategies for variance reduction. First, one may reduce the frequency at which resampling takes place. When the state is known to be static ($x_t = x_{t-1}$) one should never resample. This is the case, for example, in mobile robot localization: When the robot stops, resampling should be suspended (and in fact it is usually a good idea to suspend the integration of measurements as well). Even if the state changes, it is often a good idea to reduce the frequency of resampling. Multiple measurements can always be integrated via multiplicatively

```
1:   Algorithm Low_variance_sampler(X_t, W_t):
2:       X̄_t = ∅
3:       r = rand(0; M^{-1})
4:       c = w_t^[1]
5:       i = 1
6:       for m = 1 to M do
7:           u = r + (m − 1) · M^{-1}
8:           while u > c
9:               i = i + 1
10:              c = c + w_t^[i]
11:          endwhile
12:          add x_t^[i] to X̄_t
13:      endfor
14:      return X̄_t
```

Table 4.4 Low variance resampling for the particle filter. This routine uses a single random number to sample from the particle set $\mathcal{X}$ with associated weights $\mathcal{W}$, yet the probability of a particle to be resampled is still proportional to its weight. Furthermore, the sampler is efficient: Sampling M particles requires $O(M)$ time.

updating the importance factor as noted above. More specifically, it maintains the importance weight in memory and updates them as follows:

$$w_t^{[m]} \quad = \quad \begin{cases} 1 & \text{if resampling took place} \\ p(z_t \mid x_t^{[m]})\ w_{t-1}^{[m]} & \text{if no resampling took place} \end{cases} \tag{4.37}$$

The choice of when to resample is intricate and requires practical experience: Resampling too often increases the risk of losing diversity. If one samples too infrequently, many samples might be wasted in regions of low probability. A standard approach to determining whether or not resampling should be performed is to measure the variance of the importance weights. The variance of the weights relates to the efficiency of the sample based representation. If all weights are identical, then the variance is zero and no resampling should be performed. If, on the other hand, the weights are concentrated on a small number of samples, then the weight variance is high and resampling should be performed.

The second strategy for reducing the sampling error is known as *low variance sampling*. Table 4.4 depicts an implementation of a low variance sampler. The basic idea is that instead of selecting samples independently of each other in the

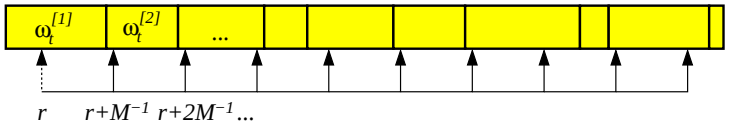

Figure 4.3 Principle of the low variance resampling procedure. We choose a random number r and then select those particles that correspond to $u = r + (m-1) \cdot M^{-1}$ where $m = 1, \ldots, M$.

resampling process (as is the case for the basic particle filter in Table 4.3), the selection involves a sequential stochastic process.

Instead of choosing M random numbers and selecting those particles that correspond to these random numbers, this algorithm computes a single random number and selects samples according to this number but still with a probability proportional to the sample weight. This is achieved by drawing a random number r in the interval $[0; M^{-1}[$, where M is the number of samples to be drawn at time t. The algorithm in Table 4.4 then selects particles by repeatedly adding the fixed amount M^{-1} to r and by choosing the particle that corresponds to the resulting number. Any number u in $[0; 1]$ points to exactly one particle, namely the particle i for which

$$i \quad = \quad \operatorname*{argmin}_j \sum_{m=1}^{j} w_t^{[m]} \geq u \tag{4.38}$$

The while loop in Table 4.4 serves two tasks, it computes the sum in the right-hand side of this equation and additionally checks whether i is the index of the first particle such that the corresponding sum of weights exceeds u. The selection is then carried out in Line 12. This process is also illustrated in Figure 4.3.

The advantage of the low-variance sampler is threefold. First, it covers the space of samples in a more systematic fashion than the independent random sampler. This should be obvious from the fact that the dependent sampler cycles through all particles systematically, rather than choosing them independently at random. Second, if all the samples have the same importance factors, the resulting sample set $\bar{\mathcal{X}}_t$ is equivalent to $\mathcal{X}_t$ so that no samples are lost if we resample without having integrated an observation into $\mathcal{X}_t$. Third, the low-variance sampler has a complexity of $O(M)$. Achieving the same complexity for independent sampling is difficult; obvious implementations require a $O(\log M)$ search for each particle once a random number has been drawn, which results in a complexity of $O(M \log M)$ for the entire resampling process. Computation time is of essence when using particle filters, and often an efficient implementation of the resampling process can make a huge difference in the practical performance. For these

reasons, most implementations of particle filters in robotics tend to rely on mechanisms like the one just discussed.

In general, the literature on efficient sampling is huge. Another popular option is *stratified sampling*, in which particles are grouped into subsets. The number of samples in each subset can be kept the same over time, regardless of the total weight of the particles contained in each subset. Such techniques tend to perform well when a robot tracks multiple, distinct hypotheses with a single particle filter.

3. A third source of error pertains to the divergence of the proposal and target distribution. We already hinted at the problem above, when discussing importance sampling. In essence, particles are generated from a proposal distribution that does not consider the measurement (cf., Equation (4.28)). The target distribution, which is the familiar Bayes filter posterior, depends of course on the measurement. The efficiency of the particle filter relies crucially on the 'match' between the proposal and the target distribution. If, at one extreme, the sensors of the robot are highly inaccurate but its motion is very accurate, the target distribution will be similar to the proposal distribution and the particle filter will be efficient. If, on the other hand, the sensors are highly accurate but the motion is not, these distributions can deviate substantially and the resulting particle filter can become arbitrarily inefficient. An extreme example of this would be a robot with *deterministic* sensors. For most deterministic sensors, the support of the measurement probability $p(z \mid x)$ will be limited to a submanifold of the state space. For example, consider a mobile robot that performs localization with noise-free range sensors. Clearly, $p(z \mid x)$ will be zero for almost every state x, with the exceptions of those that match the range measurement z exactly. Such a situation can be fatal: the proposal distribution will practically never generate a sample x which *exactly* corresponds to the range measurement z. Thus, all importance weights will be zero with probability one, and the resampling step becomes ill-conditioned. More generally, if $p(z \mid x)$ is degenerate, meaning that its support is restricted to a manifold of a smaller dimension than the dimension of the state space, the plain particle filter algorithm is inapplicable.

   There exist a range of techniques for overcoming this problem. One simple-minded technique is to simply assume more noise in perception than there actually is. For example, one might use a measurement model $p(z \mid x)$ that overestimates the actual noise in the range measurements. In many implementations, such a step improves the accuracy of the particle filter—despite the oddity of using a knowingly incorrect measurement probability. Other techniques involve modifications of the proposal distribution in ways that incorporate the measurement. Such techniques will be discussed in later chapters of this book.

4. A fourth and final disadvantage of the particle filter is known as the *particle deprivation problem*. When performing estimation in a high-dimensional space,

there may be no particles in the vicinity to the correct state. This might be because the number of particles is too small to cover all relevant regions with high likelihood. However, one might argue that this ultimately must happen in any particle filter, regardless of the particle set size M. Particle deprivation occurs as the result of random resampling; an unlucky series of random numbers can wipe out all particles near the true state. At each resampling step, the probability for this to happen is larger than zero (although it is usually exponentially small in M). Thus, we only have to run the particle filter long enough. Eventually, we will generate an estimate that is arbitrarily incorrect.

In practice, problems of this nature only tend to arise when M is small relative to the space of all states with high likelihood. A popular solution to this problem is to add a small number of randomly generated particles into the set after each resampling process, regardless of the actual sequence of motion and measurement commands. Such a methodology can reduce (but not fix) the particle deprivation problem, but at the expense of an incorrect posterior estimate. The advantage of adding random samples lies in its simplicity: The software modification necessary to add random samples in a particle filter is minimal. As a rule of thumb, adding random samples should be considered a measure of last resort, which should only be applied if all other techniques for fixing a deprivation problem have failed.

This discussion showed that the quality of the sample based representation increases with the number of samples. An important question is therefore how many samples should be used for a specific estimation problem. Unfortunately, there is no perfect answer to this question and it is often left to the user to determine the required number of samples. As a rule of thumb, the number of samples strongly depends on the dimensionality of the state space and the uncertainty of the distributions approximated by the particle filter. For example, uniform distributions require many more samples than distributions focused on a small region of the state space. A more detailed discussion on sample sizes will be given in the context of robot localization, when we consider adaptive particle filters (see Section **??**).

4.3 SUMMARY

This section introduced two nonparametric Bayes filters, histogram filters and particle filters. Nonparametric filters approximate the posterior by a finite number of values. Under mild assumptions on the system model and the shape of the posterior, both have the property that the approximation error converges uniformly to zero as the the number of values used to represent the posterior goes to infinity.

- The histogram filter decomposes the state space into finitely many convex regions. It represents the cumulative posterior probability of each region by a single numerical value.
- There exist many decomposition techniques in robotics. In particular, the granularity of a decomposition may or may not depend on the structure of the environment. When it does, the resulting algorithms are often called "topological."
- Decomposition techniques can be divided into static and dynamic. Static decompositions are made in advance, irrespective of the shape of the belief. Dynamic decompositions rely on specifics of the robot's belief when decomposing the state space, often attempting to increase spatial resolution in proportion to the posterior probability. Dynamic decompositions tend to give better results, but they are also more difficult to implement.
- An alternative nonparametric technique is known as particle filter. Particle filters represent posteriors by a random sample of states, drawn from the posterior. Such samples are called particles. Particle filter are extremely easy to implement, and they are the most versatile of all Bayes filter algorithms represented in this book.
- Specific strategies exist to reduce the error in particle filters. Among the most popular ones are techniques for reducing the variance of the estimate that arises from the randomness of the algorithm, and techniques for adapting the number of particles in accordance with the complexity of the posterior.

The filter algorithms discussed in this and the previous chapter lay the groundwork for most probabilistic robotics algorithms discussed throughout the remainder of this book. The material presented here represents many of today's most popular algorithms and representations in probabilistic robotics.

4.4 BIBLIOGRAPHICAL REMARKS