2

RECURSIVE STATE ESTIMATION

2.1 INTRODUCTION

At the core of probabilistic robotics is the idea of estimating state from sensor data. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred. In most robotic applications, determining what to do is relatively easy if one only knew *certain* quantities. For example, moving a mobile robot is relatively easy if the exact location of the robot and all nearby obstacles are known. Unfortunately, these variables are not directly measurable. Instead, a robot has to rely on its sensors to gather this information. Sensors carry only partial information about those quantities, and their measurements are corrupted by noise. State estimation seeks to recover state variables from the data. Probabilistic state estimation algorithms compute belief distributions over possible world states. An example of probabilistic state estimation was already encountered in the introduction to this book: mobile robot localization.

The goal of this chapter is to introduce the basic vocabulary and mathematical tools for estimating state from sensor data.

- Section 2.2 introduces basic probabilistic concepts and notations used throughout the book.
- Section 2.3 describes our formal model of robot environment interaction, setting forth some of the key terminology used throughout the book.
- Section 2.4 introduces *Bayes filters*, the recursive algorithm for state estimation that forms the basis of virtually every technique presented in this book.

- Section 2.5 discusses representational and computational issues that arise when implementing Bayes filters.

2.2 BASIC CONCEPTS IN PROBABILITY

This section familiarizes the reader with the basic notation and probabilistic facts and notation used throughout the book. In probabilistic robotics, quantities such as sensor measurements, controls, and the states a robot and its environment might assume are all modeled as random variables. Random variables can take on multiple values, and they do so according to specific probabilistic laws. Probabilistic inference is the process of calculating these laws for random variables that are derived from other random variables, such as those modeling sensor data.

Let X denote a random variable and x denote a specific event that X might take on. A standard example of a random variable is that of a coin flip, where X can take on the values head or tail. If the space of all values that X can take on is discrete, as is the case if X is the outcome of a coin flip, we write

$$p(X = x) \tag{2.1}$$

to denote the probability that the random variable X has value x. For example, a fair coin is characterized by $p(X = \text{head}) = p(X = \text{tail}) = \frac{1}{2}$. Discrete probabilities sum to one, that is,

$$\sum_x p(X = x) \quad = \quad 1\,, \tag{2.2}$$

and of course, probabilities are always non-negative, that is, $p(X = x) \geq 0$. To simplify the notation, we will usually omit explicit mention of the random variable whenever possible, and instead use the common abbreviation $p(x)$ instead of writing $p(X = x)$.

Most techniques in this book address estimation and decision making in continuous spaces. Continuous spaces are characterized by random variables that can take on a continuum of values. Throughout this book, we assume that all continuous random variables possess probability density functions (PDFs). A common density function is that of the one-dimensional *normal distribution* with mean μ and variance σ^2. This

distribution is given by the following Gaussian function:

$$p(x) \quad = \quad \left(2\pi\sigma^2\right)^{-\frac{1}{2}} \; \exp\left\{-\tfrac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\} \tag{2.3}$$

Normal distributions play a major role in this book. We will frequently abbreviate them as $\mathcal{N}(x;\mu,\sigma^2)$, which specifies the random variable, its mean, and its variance.

The Normal distribution (2.3) assumes that x is a scalar value. Often, x will be a multi-dimensional vector. Normal distributions over vectors are called *multivariate*. Multivariate normal distributions are characterized by density functions of the following form:

$$p(x) \quad = \quad \det\left(2\pi\Sigma\right)^{-\frac{1}{2}} \; \exp\left\{-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{2.4}$$

Here μ is the mean vector and Σ a (positive semidefinite) symmetric matrix called *covariance matrix*. The superscript ^{T} marks the transpose of a vector. The reader should take a moment to realize that Equation (2.4) is a strict generalization of Equation (2.3); both definitions are equivalent if x is a scalar value. The PDFs of a one- and a two-dimensional normal distribution are graphically depicted in Figure 5.6.

Equations (2.3) and (2.4) are examples of PDFs. Just as discrete probability distributions always sums up to one, a PDF always integrates to 1:

$$\int p(x)\; dx \quad = \quad 1\;. \tag{2.5}$$

However, unlike a discrete probability, the value of a PDF is not bounded above by 1. Throughout this book, we will use the terms *probability*, *probability density* and *probability density function* interchangeably. We will silently assume that all continuous random variables are measurable, and we also assume that all continuous distributions actually possess densities.

The *joint distribution* of two random variables X and Y is given by

$$p(x,y) \quad = \quad p(X = x \text{ and } Y = y)\;. \tag{2.6}$$

This expression describes the probability of the event that the random variable X takes on the value x *and* that Y takes on the value y. If X and Y are *independent*, we have

$$p(x, y) \quad = \quad p(x)\, p(y)\ . \tag{2.7}$$

Often, random variables carry information about other random variables. Suppose we already know that Y's value is y, and we would like to know the probability that X's value is x conditioned on that fact. Such a probability will be denoted

$$p(x \mid y) \quad = \quad p(X = x \mid Y = y) \tag{2.8}$$

and is called *conditional* probability. If $p(y) > 0$, then the conditional probability is defined as

$$p(x \mid y) \quad = \quad \frac{p(x, y)}{p(y)}\ . \tag{2.9}$$

If X and Y are independent, we have

$$p(x \mid y) \quad = \quad \frac{p(x)\, p(y)}{p(y)} \quad = \quad p(x)\ . \tag{2.10}$$

In other words, if X and Y are independent, Y tells us nothing about the value of X. There is no advantage of knowing Y if our interest pertains to knowing X. Independence, and its generalization known as conditional independence, plays a major role throughout this book.

An interesting fact, which follows from the definition of conditional probability and the axioms of probability measures, is often referred to as *theorem of total probability*:

$$p(x) \quad = \quad \sum_{y} p(x \mid y)\, p(y) \qquad \text{(discrete case)} \tag{2.11}$$

$$p(x) \quad = \quad \int p(x \mid y)\, p(y)\, dy \qquad \text{(continuous case)} \tag{2.12}$$

If $p(x \mid y)$ or $p(y)$ are zero, we define the product $p(x \mid y)\ p(y)$ to be zero, regardless of the value of the remaining factor.

Equally important is *Bayes rule*, which relates conditionals of the type $p(x \mid y)$ to their "inverse," $p(y \mid x)$. The rule, as stated here, requires $p(y) > 0$:

$$p(x \mid y) = \frac{p(y \mid x)\, p(x)}{p(y)} = \frac{p(y \mid x)\, p(x)}{\sum_{x'} p(y \mid x')\, p(x')} \quad \text{(discrete)} \tag{2.13}$$

$$p(x \mid y) = \frac{p(y \mid x)\, p(x)}{p(y)} = \frac{p(y \mid x)\, p(x)}{\int p(y \mid x')\, p(x')\, dx'} \quad \text{(continuous)} \tag{2.14}$$

Bayes rule plays a predominant role in probabilistic robotics. If x is a quantity that we would like to infer from y, the probability $p(x)$ will be referred to as *prior probability distribution*, and y is called the *data* (e.g., a sensor measurement). The distribution $p(x)$ summarizes the knowledge we have regarding X prior to incorporating the data y. The probability $p(x \mid y)$ is called the *posterior probability distribution* over X. As (2.14) suggests, Bayes rule provides a convenient way to compute a posterior $p(x \mid y)$ using the "inverse" conditional probability $p(y \mid x)$ along with the prior probability $p(x)$. In other words, if we are interested in inferring a quantity x from sensor data y, Bayes rule allows us to do so through the inverse probability, which specifies the probability of data y assuming that x was the case. In robotics, this inverse probability is often coined "generative model," since it describes, at some level of abstraction, how state variables X *cause* sensor measurements Y.

An important observation is that the denominator of Bayes rule, $p(y)$, does not depend on x. Thus, the factor $p(y)^{-1}$ in Equations (2.13) and (2.14) will be the same for any value x in the posterior $p(x \mid y)$. For this reason, $p(y)^{-1}$ is often written as a *normalizer* variable, and generically denoted η:

$$p(x \mid y) = \eta\, p(y \mid x)\, p(x)\,. \tag{2.15}$$

If X is discrete, equations of this type can be computed as follows:

$$\forall x : \mathrm{aux}_{x|y} = p(y \mid x)\, p(x) \tag{2.16}$$

$$\mathrm{aux}_{y} = \sum_{x} \mathrm{aux}_{x|y} \tag{2.17}$$

$$\forall x : p(x \mid y) = \frac{\mathrm{aux}_{x|y}}{\mathrm{aux}_{y}}\,, \tag{2.18}$$

where $\mathrm{aux}_{x|y}$ and aux_{y} are auxiliary variables. These instructions effectively calculate $p(x \mid y)$, but instead of explicitly computing $p(y)$, they instead just normalize the

result. The advantage of the notation in (2.15) lies in its brevity. Instead of explicitly providing the exact formula for a normalization constant—which can grow large very quickly in some of the mathematical derivations to follow—we simply will use the normalizer "η" to indicate that the final result has to be normalized to 1. Throughout this book, normalizers of this type will be denoted η (or η', η'', ...). We will freely use the same η in different equations to denote normalizers, even if their actual values are different.

The *expectation* of a random variable X is given by

$$\begin{aligned} E[X] &= \sum_x x\ p(x)\ , \\ E[X] &= \int x\ p(x)\ dx\ . \end{aligned} \tag{2.19}$$

Not all random variables possess finite expectations; however, those that do not are of no relevance to the material presented in this book. The expectation is a linear function of a random variable. In particular, we have

$$E[aX + b] = aE[X] + b \tag{2.20}$$

for arbitrary numerical values a and b. The covariance of X is obtained as follows

$$\mathrm{Cov}[X] = E[X - E[X]]^2 = E[X^2] - E[X]^2 \tag{2.21}$$

The covariance measures the squared expected deviation from the mean. As stated above, the mean of a multivariate normal distribution $\mathcal{N}(x; \mu, \Sigma)$ is μ, and its covariance is Σ.

Another important characteristic of a random variable is its *entropy*. For discrete random variables, the entropy is given by the following expression:

$$H(P) = E[-\log_2 p(x)] = -\sum_x p(x)\ \log_2 p(x)\ . \tag{2.22}$$

The concept of entropy originates in information theory. The entropy is the expected information that the value of x carries: $-\log_2 p(x)$ is the number of bits required

to encode x using an optimal encoding, and $p(x)$ is the probability at which x will be observed. In this book, entropy will be used in robotic information gathering, to express the information a robot may receive upon executing specific actions.

Finally, we notice that it is perfectly fine to condition any of the rules discussed so far on arbitrary other random variables, such as the variable Z. For example, conditioning Bayes rule on $Z = z$ gives us:

$$p(x \mid y, z) \quad = \quad \frac{p(y \mid x, z)\, p(x \mid z)}{p(y \mid z)} \tag{2.23}$$

Similarly, we can condition the rule for combining probabilities of independent random variables (2.7) on other variables z:

$$p(x, y \mid z) \quad = \quad p(x \mid z)\, p(y \mid z)\,. \tag{2.24}$$

Such a relation is known as *conditional independence*. As the reader easily verifies, (2.24) is equivalent to

$$\begin{aligned} p(x \mid z) \quad &= \quad p(x \mid z, y) \\ p(y \mid z) \quad &= \quad p(y \mid z, x) \end{aligned} \tag{2.25}$$

Conditional independence plays an important role in probabilistic robotics. It applies whenever a variable y carries no information about a variable x if another variable's value z is known. Conditional independence does not imply (absolute) independence, that is,

$$p(x, y \mid z) \;=\; p(x \mid z)\, p(y \mid z) \quad \not\Rightarrow \quad p(x, y) \;=\; p(x)\, p(y) \tag{2.26}$$

The converse is also in general untrue: absolute independence does not imply conditional independence:

$$p(x, y) \;=\; p(x)\, p(y) \quad \not\Rightarrow \quad p(x, y \mid z) \;=\; p(x \mid z)\, p(y \mid z) \tag{2.27}$$

In special cases, however, conditional and absolute independence may coincide.

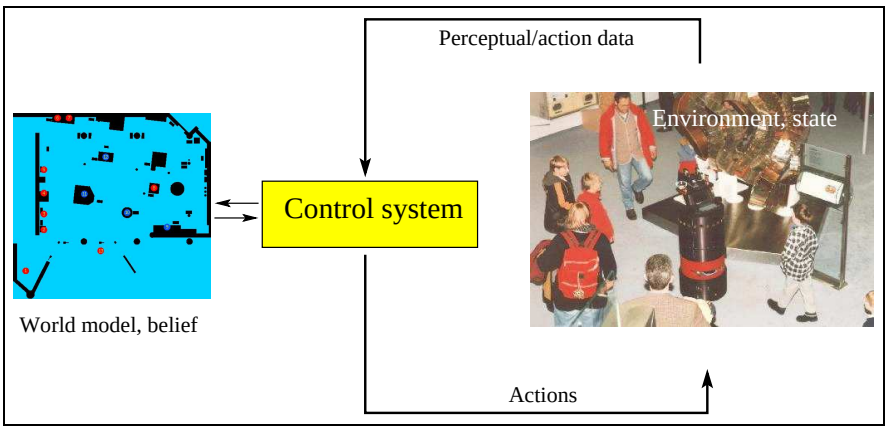

Figure 2.1 Robot Environment Interaction.

2.3 ROBOT ENVIRONMENT INTERACTION

Figure 2.1 illustrates the interaction of a robot with its environment. The *environment*, or *world*, of a robot is a dynamical system that possesses internal state. The robot can acquire information about its environment using its sensors. However, sensors are noisy, and there are usually many things that cannot be sensed directly. As a consequence, the robot maintains an internal belief with regards to the state of its environment, depicted on the left in this figure. The robot can also influence its environment through its actuators. However, the effect of doing so is often somewhat unpredictable. This interaction will now be described more formally.

2.3.1 State

Environments are characterized by *state*. For the material presented in this book, it will be convenient to think of state as the collection of all aspects of the robot and its environment that can impact the future. State may change over time, such as the location of people; or it may remain static throughout the robot's operation, such as the location of walls in (most) buildings. State that changes will be called *dynamic state*, which distinguishes it from *static*, or non-changing state. The state also includes variables regarding the robot itself, such as its pose, velocity, whether or not its sensors are functioning correctly, and so on. Throughout this book, state will be denoted x; although the specific variables included in x will depend on the context. The state at time t will be denoted x_t. Typical state variables used throughout this book are:

- The robot pose, which comprises its location and orientation relative to a global coordinate frame. Rigid mobile robots possess six such state variables, three for their Cartesian coordinates, and three for their angular orientation, also called Euler angles (pitch, roll, and yaw). For rigid mobile robots confined to planar environments, the pose is usually given by three variables, its two location coordinates in the plane and its heading direction (yaw). The robot pose is often referred to as *kinematic state*.
- The configuration of the robot's actuators, such as the joints of robotic manipulators. Each degree of freedom in a robot arm is characterized by a one-dimensional configuration at any point in time, which is part of the kinematic state of the robot.
- The robot velocity and the velocities of its joints. A rigid robot moving through space is characterized by up to six velocity variables, one for each pose variables. Velocities are commonly referred to as *dynamic state*. Dynamic state will play only a minor role in this book.
- The location and features of surrounding objects in the environment. An object may be a tree, a wall, or a pixel within a larger surface. Features of such objects may be their visual appearance (color, texture). Depending on the granularity of the state that is being modeled, robot environments possess between a few dozen and up to hundreds of billions of state variables (and more). Just imagine how many bits it will take to accurately describe your physical environment! For many of the problems studied in this book, the location of objects in the environment will be static. In some problems, objects will assume the form of *landmarks*, which are distinct, stationary features of the environment that can be recognized reliably.
- The location and velocities of moving objects and people. Often, the robot is not the only moving actor in its environment. Other moving entities possess their own kinematic and dynamic state.
- There can be a huge number of other state variables. For example, whether or not a sensor is broken is a state variable, as is the level of battery charge for a battery-powered robot.

A state x_t will be called *complete* if it is the best predictor of the future. Put differently, completeness entails that knowledge of past states, measurements, or controls carry no additional information that would help us to predict the future more accurately. It it important to notice that our definition of completeness does not require the future to be a *deterministic* function of the state. The future may be stochastic, but no variables prior to x_t may influence the stochastic evolution of future states, unless

this dependence is mediated through the state x_t. Temporal processes that meet these conditions are commonly known as *Markov chains*.

The notion of state completeness is mostly of theoretical importance. In practice, it is impossible to specify a complete state for any realistic robot system. A complete state includes not just all aspects of the environment that may have an impact on the future, but also the robot itself, the content of its computer memory, the brain dumps of surrounding people, etc. Those are hard to obtain. Practical implementations therefore single out a small subset of all state variables, such as the ones listed above. Such a state is called *incomplete state*.

In most robotics applications, the state is continuous, meaning that x_t is defined over a continuum. A good example of a continuous state space is that of a robot pose, that is, its location and orientation relative to an external coordinate system. Sometimes, the state is discrete. An example of a discrete state space is the (binary) state variable that models whether or not a sensor is broken. State spaces that contain both continuous and discrete variables are called *hybrid* state spaces.

In most cases of interesting robotics problems, state changes over time. Time, throughout this book, will be discrete, that is, all interesting events will take place at discrete time steps

$$0, 1, 2, \ldots \ . \tag{2.28}$$

If the robot starts its operation at a distinct point in time, we will denote this time as $t = 0$.

2.3.2 Environment Interaction

There are two fundamental types of interactions between a robot and its environment: The robot can influence the state of its environment through its actuators. And it can gather information about the state through its sensors. Both types of interactions may co-occur, but for didactic reasons we will distinguish them throughout this book. The interaction is illustrated in Figure 2.1:

- **Sensor measurements**. Perception is the process by which the robot uses its sensors to obtain information about the state of its environment. For example, a robot might take a camera image, a range scan, or query its tactile sensors to receive information about the state of the environment. The result of such a

perceptual interaction will be called a *measurement*, although we will sometimes also call it *observation* or *percept*. Typically, sensor measurements arrive with some delay. Hence they provide information about the state a few moments ago.

- **Control actions** change the state of the world. They do so by actively asserting forces on the robot's environment. Examples of control actions include robot motion and the manipulation of objects. Even if the robot does not perform any action itself, state usually changes. Thus, for consistency, we will assume that the robot *always* executes a control action, even if it chooses not to move any of its motors. In practice, the robot continuously executes controls and measurements are made concurrently.

Hypothetically, a robot may keep a record of all past sensor measurements and control actions. We will refer to such a the collection as the *data* (regardless of whether they are being memorized). In accordance with the two types of environment interactions, the robot has access to two different data streams.

- **Measurement data** provides information about a momentary state of the environment. Examples of measurement data include camera images, range scans, and so on. For most parts, we will simply ignore small timing effects (e.g., most ladar sensors scan environments sequentially at very high speeds, but we will simply assume the measurement corresponds to a specific point in time). The measurement data at time t will be denoted

$$z_t \tag{2.29}$$

  Throughout most of this book, we simply assume that the robot takes exactly one measurement at a time. This assumption is mostly for notational convenience, as nearly all algorithms in this book can easily be extended to robots that can acquire variables numbers of measurements within a single time step. The notation

$$z_{t_1:t_2} \quad = \quad z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2} \tag{2.30}$$

  denotes the set of all measurements acquired from time t_1 to time t_2, for $t_1 \leq t_2$.

- **Control data** carry information about the *change of state* in the environment. In mobile robotics, a typical example of control data is the velocity of a robot. Setting the velocity to 10 cm per second for the duration of five seconds suggests that the robot's pose, after executing this motion command, is approximately 50 cm ahead of its pose before command execution. Thus, its main information regards the change of state.

An alternative source of control data are *odometers*. Odometers are sensors that measure the revolution of a robot's wheels. As such they convey information about the change of the state. Even though odometers are sensors, we will treat odometry as control data, since its main information regards the change of the robot's pose.

Control data will be denoted u_t. The variable u_t will always correspond to the change of state in the time interval $(t-1;t]$. As before, we will denote sequences of control data by $u_{t_1:t_2}$, for $t_1 \leq t_2$:

$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2} . \tag{2.31}$$

Since the environment may change even if a robot does not execute a specific control action, the fact that time passed by constitutes, technically speaking, control information. Hence, we assume that there is exactly one control data item per time step t.

The distinction between measurement and control is a crucial one, as both types of data play fundamentally different roles in the material yet to come. Perception provides information about the environment's state, hence it tends to increase the robot's knowledge. Motion, on the other hand, tends to induce a loss of knowledge due to the inherent noise in robot actuation and the stochasticity of robot environments; although sometimes a control makes the robot more certain about the state. By no means is our distinction intended to suggest that actions and perceptions are separated in time, i.e., that the robot does not move while taking sensor measurements. Rather, perception and control takes place concurrently; many sensors affect the environment; and the separation is strictly for convenience.

2.3.3 Probabilistic Generative Laws

The evolution of state and measurements is governed by probabilistic laws. In general, the state at time x_t is generated stochastically. Thus, it makes sense to specify the probability distribution from which x_t is generated. At first glance, the emergence of state x_t might be conditioned on all past states, measurements, and controls. Hence, the probabilistic law characterizing the evolution of state might be given by a probability distribution of the following form:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \tag{2.32}$$

(Notice that through no particular motivation we assume here that the robot executes a control action u_1 first, and then takes a measurement z_1.) However, if the state x is

complete then it is a sufficient summary of all that happened in previous time steps. In particular, x_{t-1} is a sufficient statistic of all previous controls and measurements up to this point, that is, $u_{1:t-1}$ and $z_{1:t-1}$. From all the variables in the expression above, only the control u_t matters if we know the state x_{t-1}. In probabilistic terms, this insight is expressed by the following equality:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \quad = \quad p(x_t \mid x_{t-1}, u_t) \tag{2.33}$$

The property expressed by this equality is an example of *conditional independence*. It states that certain variables are independent of others if one knows the values of a third group of variables, the conditioning variables. Conditional independence will be exploited pervasively in this book, as it is the main source of tractability of probabilistic robotics algorithms.

Similarly, one might want to model the process by which measurements are being generated. Again, if x_t is complete, we have an important conditional independence:

$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) \quad = \quad p(z_t \mid x_t) \tag{2.34}$$

In other words, the state x_t is sufficient to predict the (potentially noisy) measurement z_t. Knowledge of any other variable, such as past measurements, controls or even past states, is irrelevant if x_t is complete.

This discussion leaves open as to what the two resulting conditional probabilities are: $p(x_t \mid x_{t-1}, u_t)$ and $p(z_t \mid x_t)$. The probability $p(x_t \mid x_{t-1}, u_t)$ is the *state transition probability*. It specifies how environmental state evolves over time as a function of robot controls u_t. Robot environments are stochastic, which is reflected by the fact that $p(x_t \mid x_{t-1}, u_t)$ is a probability distribution, not a deterministic function. Sometimes the state transition distribution does not depend on the time index t, in which case we may write it as $p(x' \mid u, x)$, where x' is the successor and x the predecessor state.

The probability $p(z_t \mid x_t)$ is called the *measurement probability*. It also may not depend on the time index t, in which case it shall be written as $p(z \mid x)$. The measurement probability specifies the probabilistic law according to which measurements z are generated from the environment state x. Measurements are usually noisy projections of the state.

The state transition probability and the measurement probability together describe the dynamical stochastic system of the robot and its environment. Figure **??** illustrates the evolution of states and measurements, defined through those probabilities. The

state at time t is stochastically dependent on the state at time $t-1$ and the control u_t. The measurement z_t depends stochastically on the state at time t. Such a temporal generative model is also known as hidden Markov model (HMM) or dynamic Bayes network (DBN). To specify the model fully, we also need an initial state distribution $p(x_0)$.

2.3.4 Belief Distributions

Another key concept in probabilistic robotics is that of a *belief*. A belief reflects the robot's internal knowledge about the state of the environment. We already discussed that state cannot be measured directly. For example, a robot's pose might be $x = \langle 14.12, 12.7, 0.755 \rangle$ in some global coordinate system, but it usually cannot know its pose, since poses are not measurable directly (not even with GPS!). Instead, the robot must infer its pose from data. We therefore distinguish the true state from its internal *belief*, or *state of knowledge* with regards to that state.

Probabilistic robotics represents beliefs through conditional probability distributions. A belief distribution assigns a probability (or density value) to each possible hypothesis with regards to the true state. Belief distributions are posterior probabilities over state variables conditioned on the available data. We will denote belief over a state variable x_t by $bel(x_t)$, which is an abbreviation for the posterior

$$bel(x_t) \quad = \quad p(x_t \mid z_{1:t}, u_{1:t}) \; . \tag{2.35}$$

This posterior is the probability distribution over the state x_t at time t, conditioned on all past measurements $z_{1:t}$ and all past controls $u_{1:t}$.

The reader may notice that we silently assume that the belief is taken *after* incorporating the measurement z_t. Occasionally, it will prove useful to calculate a posterior *before* incorporating z_t, just after executing the control u_t. Such a posterior will be denoted as follows:

$$\overline{bel}(x_t) \quad = \quad p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{2.36}$$

This probability distribution is often referred to as *prediction* in the context of probabilistic filtering. This terminology reflects the fact that $\overline{bel}(x_t)$ predicts the state at time t based on the previous state posterior, before incorporating the measurement at time t. Calculating $bel(x_t)$ from $\overline{bel}(x_t)$ is called *correction* or the *measurement update*.

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:

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

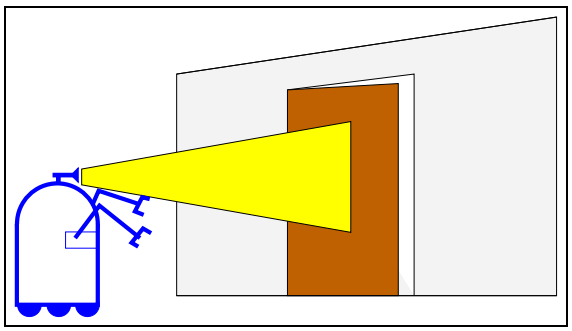

Figure 2.2 A mobile robot estimating the state of a door.

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