1 Introduction

1.1 Uncertainty in Robotics

Robotics is the science of perceiving and manipulating the physical world through computer-controlled devices. Examples of successful robotic systems include mobile platforms for planetary exploration, industrial robotics arms in assembly lines, cars that travel by themselves, and manipulators that assist surgeons. Robotics systems are situated in the physical world, perceive information on their environments through sensors, and manipulate through physical forces.

While much of robotics is still in its infancy, the idea of "intelligent" manipulating devices has an enormous potential to change society. Wouldn't it be great if all our cars were able to safely steer themselves, making car accidents a notion of the past? Wouldn't it be great if robots, and not people, would clean up nuclear disaster sites like Chernobyl? Wouldn't it be great if our homes were populated by intelligent assistants that take care of all domestic repair and maintenance tasks?

To do these tasks, robots have to be able to accommodate the enormous uncertainty that exists in the physical world. There is a number of factors that contribute to a robot's uncertainty.

First and foremost, robot environments are inherently unpredictable. While the degree of uncertainty in well-structured environments such as assembly lines is small, environments such as highways and private homes are highly dynamic and in many ways highly unpredictable. The uncertainty is particularly high for robots operating in the proximity of people.

Sensors are limited in what they can perceive. Limitations arise from several factors. The range and resolution of a sensor is subject to physical limitations. For example, cameras cannot see through walls, and the spatial res-
olution of a camera image is limited. Sensors are also subject to noise, which perturbs sensor measurements in unpredictable ways and hence limits the information that can be extracted. And finally, sensors can break. Detecting a faulty sensor can be extremely difficult.

Robot actuation involves motors that are, at least to some extent, unpredictable. Uncertainty arises from effects like control noise, wear-and-tear, and mechanical failure. Some actuators, such as heavy-duty industrial robot arms, are quite accurate and reliable. Others, like low-cost mobile robots, can be extremely flaky.

Some uncertainty is caused by the robot’s software. All internal models of the world are approximate. Models are abstractions of the real world. As such, they only partially model the underlying physical processes of the robot and its environment. Model errors are a source of uncertainty that has often been ignored in robotics, despite the fact that most robotic models used in state-of-the-art robotic systems are rather crude.

Model errors are a source of uncertainty that has often been ignored in robotics, despite the fact that most robotic models used in state-of-the-art robotic systems are rather crude.

Uncertainty is further created through algorithmic approximations. Robots are real-time systems. This limits the amount of computation that can be carried out. Many popular algorithms are approximate, achieving timely response through sacrificing accuracy.

The level of uncertainty depends on the application domain. In some robotic applications, such as assembly lines, humans can cleverly engineer the system so that uncertainty is only a marginal factor. In contrast, robots operating in residential homes or on other planets will have to cope with substantial uncertainty. Such robots are forced to act even though neither their sensors, nor their internal models, will provide it with sufficient information to make the right decisions with absolute certainty. As robotics is now moving into the open world, the issue of uncertainty has become a major stumbling block for the design of capable robot systems. Managing uncertainty is possibly the most important step towards robust real-world robot systems.

Hence this book.

1.2 Probabilistic Robotics

This book provides a comprehensive overview of probabilistic robotics. Probabilistic robotics is a relatively new approach to robotics that pays tribute to the uncertainty in robot perception and action. The key idea in probabilistic robotics is to represent uncertainty explicitly using the calculus of probability theory. Put differently, instead of relying on a single “best guess” as to what might be the case, probabilistic algorithms represent information by probability distributions over a whole space of guesses. By doing so, they can represent ambiguity and degree of belief in a mathematically sound way.

Control choices can be made robust relative the uncertainty that remains, and probabilistic robotics can even actively chose to reduce their uncertainty when this appears to be the superior choice. Thus, probabilistic algorithms degrade gracefully in the face of uncertainty. As a result, they outperform alternative techniques in many real-world applications.

We shall illustrate probabilistic robotics with two motivating examples: one pertaining to robot perception, and another to planning and control.

Our first example is mobile robot localization. Robot localization is the problem of estimating a robot’s coordinates relative to an external reference frame. The robot is given a map of its environment, but to localize itself relative to this map it needs to consult its sensor data. Figure 1.1 illustrates such a situation. The environment is known to possess three indistinguishable doors. The task of the robot is to find out where it is, through sensing and motion.

This specific localization problem is known as global localization. In global localization, a robot is placed somewhere in a known environment and has to localize itself from scratch. The probabilistic paradigm represents the robot’s momentary belief by a probability density function over the space of all locations. This is illustrated in diagram (a) in Figure 1.1. This diagram shows a uniform distribution over all locations. Now suppose the robot takes a first sensor measurement and observes that it is next to a door. Probabilistic techniques exploit this information to update the belief. The ‘posterior’ belief is shown in diagram (b) in Figure 1.1. It places an increased probability at places near doors, and lower probability near walls. Notice that this distribution possesses three peaks, each corresponding to one of the indistinguishable doors in the environment. Thus, by no means does the robot know where it is. Instead, it now has three, distinct hypotheses which are each equally plausible given the sensor data. We also note that the robot assigns positive probability to places not next to a door. This is the natural result of the inherent uncertainty in sensing: With a small, non-zero probability, the robot might have erred in its assessment of seeing a door. The ability to maintain low-probability hypotheses is essential for attaining robustness.

Now suppose the robot moves. Diagram (c) in Figure 1.1 shows the effect on a robot’s belief. The belief has been shifted in the direction of motion. It also possesses a larger spread, which reflects the uncertainty that is intro-
1 Introduction

Figure 1.1 The basic idea of Markov localization. A mobile robot during global localization. Markov localization techniques will be investigated in Chapters 7 and 8.

1.2 Probabilistic Robotics

Figure 1.2 Top image: a robot navigating through open, featureless space may lose track of where it is. Bottom: This can be avoided by staying near known obstacles. These figures are results of an algorithm called coastal navigation, which will be discussed in Chapter 16. Images courtesy of Nicholas Roy, MIT.
duced by robot motion. Diagram (d) in Figure 1.1 depicts the belief after observing another door. This observation leads our algorithm to place most of the probability mass on a location near one of the doors, and the robot is now quite confident as to where it is. Finally, Diagram (e) shows a belief as the robot travels further down the corridor.

This example illustrates many aspects of the probabilistic paradigm. Stated probabilistically, the robot perception problem is a state estimation problem, and our localization example uses an algorithm known as Bayes filter for posterior estimation over the space of robot locations. The representation of information is a probability density function. The update of this function represents the information gained through sensor measurements, or the information lost through processes in the world that increase a robot's uncertainty.

Our second example brings us into the realm of robotic planning and control. As just argued, probabilistic algorithms can compute a robot's momentary uncertainty. But they can also anticipate future uncertainty, and take such uncertainty into consideration when determining the right choice of control. One such algorithm is called coastal navigation. An example of coastal navigation is shown in Figure 1.2. This figure shows a 2-D map of an actual building. The top diagram compares an estimated path with an actual path: the divergence is the result of the uncertainty in robot motion that we just discussed. The interesting insight is: not all trajectories induce the same level of uncertainty. The path in Figure 1.2a leads through relatively open space, deprived of features that could help the robot to remain localized. Figure 1.2b shows an alternative path. This trajectory seeks a distinct corner, and then "hugs" the wall so as to stay localized. Not surprisingly, the uncertainty will be reduced for the latter path, hence chances of arriving at the goal location are actually higher.

This example illustrates one of the many ways proper consideration of uncertainty affects the robot's controls. In our example, the anticipation of possible uncertainty along one trajectory makes the robot prefer a second, longer path, just so as to reduce the uncertainty. The new path is better, in the sense that the robot has a much higher chance of actually being at the goal when believing that it is. In fact, the second path is an example of active information gathering. The robot has, through its probabilistic consideration, determined that the best choice of action is to seek information along its path, in its pursuit to reach a target location. Probabilistic planning techniques anticipate uncertainty and can plan for information gathering, and probabilistic control techniques realize the results of such plans.

1.3 Implications

Probabilistic robotics seamlessly integrates models with sensor data, overcoming the limitations of both at the same time. These ideas are not just a matter of low-level control. They cut across all levels of robotic software, from the lowest to the highest.

In contrast with traditional programming techniques in robotics—such as model-based motion planning techniques or reactive behavior-based approaches—probabilistic approaches tend to be more robust in the face of sensor limitations and model limitations. This enables them to scale much better to complex real-world environments than previous paradigms, where uncertainty is of even greater importance. In fact, certain probabilistic algorithms are currently the only known working solutions to hard robotic estimation problems, such as the localization problem discussed a few pages ago, or the problem of building accurate maps of very large environments.

In comparison to traditional model-based robotic techniques, probabilistic algorithms have weaker requirements on the accuracy of the robot's models, thereby relieving the programmer from the insurmountable burden to come up with accurate models. Probabilistic algorithms have weaker requirements on the accuracy of robotic sensors than those made by many reactive techniques, whose sole control input is the momentary sensor input. Viewed probabilistically, the robot learning problem is a long-term estimation problem. Thus, probabilistic algorithms provide a sound methodology for many flavors of robot learning.

However, these advantages come at a price. The two most frequently cited limitations of probabilistic algorithms are computational complexity, and a need to approximate. Probabilistic algorithms are inherently less efficient than their non-probabilistic counterparts. This is due to the fact that they consider entire probability densities instead of a single guess. The need to approximate arises from the fact that most robot worlds are continuous. Computing exact posterior distributions tends to be computationally intractable. Sometimes, one is fortunate in that the uncertainty can be approximated tightly with a compact parametric model (e.g., Gaussians). In other cases, such approximations are too crude to be of use, and more complicated representations must be employed.

Recent developments in computer hardware has made an unprecedented number of FLOPS available at bargain prices. This development has certainly aided the field of probabilistic robotics. Further, recent research has successfully increased the computational efficiency of probabilistic algo-
rithms, for a range of hard robotics problems—many of which are described in depth in this book. Nevertheless, computational challenges remain. We shall revisit this discussion at numerous places, where we investigate the strengths and weaknesses of specific probabilistic solutions.

1.4 Road Map

This book is organized in four major parts.

• Chapters 2 through 4 introduce the basic mathematical framework that underlies all of the algorithms described in this book, along with key algorithms. These chapters are the mathematical foundation of this book.

• Chapters 5 and 6 present probabilistic models of mobile robots. In many ways, these chapters are the probabilistic generalization of classical robotics models. They form the robotic foundation for the material that follows.

• The mobile robot localization problem is discussed in Chapters 7 and 8. These chapters combine the basic estimation algorithms with the probabilistic models discussed in the previous two chapters.

• Chapters 9 through 13 discuss the much richer problem of robotic mapping. As before, they are all based on the algorithms discussed in the foundational chapters, but many of them utilize tricks to accommodate the enormous complexity of this problem.

• Problems of probabilistic planning and control are discussed in Chapters 14 through 17. Here we begin by introducing a number of fundamental techniques, and then branch into practical algorithms for controlling a robot probabilistically. The final chapter, Chapter 17, discusses the problem of robot exploration from a probabilistic perspective.

The book is best read in order, from the beginning to the end. However, we have attempted to make each individual chapter self-explanatory. Frequent sections called "Mathematical Derivation of . . ." can safely be skipped on first reading without compromising the coherence of the overall material in this book.
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.

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

This section familiarizes the reader with the basic notation and probabilistic facts used throughout the book. In probabilistic robotics, quantities such as sensor measurements, controls, and the states of a robot and its environment 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 and the observed data.

Let $X$ denote a random variable and $x$ denote a specific value that $X$ might assume. A standard example of a random variable is a coin flip, where $X$ can take on the values heads or tails. 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

$$p(X = x)$$

(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,

$$\sum_x p(X = x) = 1$$

(2.2)

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. Continuous variables are described by the probability density function (PDF). A common density function is that of the one-dimensional normal distribution with mean $\mu$ and variance $\sigma^2$. The PDF of a normal distribution is given by the following Gaussian function:

$$p(x) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{(x - \mu)^2}{\sigma^2} \right)$$

(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:

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

(2.4)

Here $\mu$ is the mean vector. $\Sigma$ a positive semi-definite and symmetric matrix called the covariance matrix. The superscript $T$ marks the transpose of a vector. The argument in the exponent in this PDF is quadratic in $x$, and the parameters of this quadratic function are $\mu$ and $\Sigma$.

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 and $\Sigma = \sigma^2$.

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

$$\int p(x) \, dx = 1$$

(2.5)

However, unlike a discrete probability, the value of a PDF is not upper-bounded 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

$$p(x, y) = p(X = x \text{ and } Y = y)$$

(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

$$p(x, y) = p(x) p(y)$$

(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

$$p(x \mid y) = p(X = x \mid Y = y)$$

(2.8)
Theorem 2.14 suggests that Bayes rule generalizes the knowledge we have regarding the probability of total probability: the ability and the axioms of probability measures, is often referred to as 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:

\[
\begin{align*}
\text{(2.11)} & \quad p(x) = \sum_y p(x \mid y) p(y) \quad \text{(discrete case)} \\
\text{(2.12)} & \quad p(x) = \int p(x \mid y) p(y) \, dy \quad \text{(continuous case)}
\end{align*}
\]

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.

Bayes rule plays a predominant role in probabilistic robotics (and probabilistic inference in general). If \( x \) is a quantity that we would like to infer from \( y \), the probability \( p(x) \) will be referred to as the 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, the probability \( p(y \mid x) \) 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 in Bayes rule variable, and generically denoted \( \eta \).

\[
\begin{align*}
\text{(2.15)} & \quad p(x \mid y) = \eta p(y \mid x) p(x)
\end{align*}
\]

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

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

\[
\begin{align*}
\text{(2.16)} & \quad p(x \mid y, z) = \frac{p(y \mid x, z) p(x \mid z)}{p(y \mid z)}
\end{align*}
\]

as long as \( p(y \mid z) > 0 \).

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

\[
\begin{align*}
\text{(2.17)} & \quad p(x, y \mid z) = p(x \mid z) p(y \mid z)
\end{align*}
\]

Such a relation is known as conditional independence. As the reader easily verifies, (2.17) is equivalent to

\[
\begin{align*}
\text{(2.18)} & \quad p(x \mid z, y) = p(x \mid z) \\
\text{(2.19)} & \quad p(y \mid x, z) = p(y \mid z)
\end{align*}
\]

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
Recursive State Estimation

imply (absolute) independence, that is,
\begin{equation}
\Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z) 
\end{equation}
\begin{equation}
\Pr(x, y) = \Pr(x) \Pr(y) = \Pr(x \mid z) \Pr(y \mid z)
\end{equation}
The converse is also in general untrue: absolute independence does not imply conditional independence:
\begin{equation}
\Pr(x, y) = \Pr(x) \Pr(y) 
\end{equation}
\begin{equation}
\Pr(x \mid z) \Pr(y \mid z) 
\end{equation}
In special cases, however, conditional and absolute independence may coincide.

A number of probabilistic algorithms require us to compute features, or expectation, of probability distributions. The expectation of a random variable $X$ is given by
\begin{equation}
\mathbb{E}[X] = \sum_x x \Pr(x) \quad \text{(discrete)}
\end{equation}
\begin{equation}
\mathbb{E}[X] = \int x \Pr(x) \, dx \quad \text{(continuous)}
\end{equation}
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
\begin{equation}
\mathbb{E}[aX + b] = a \mathbb{E}[X] + b
\end{equation}
for arbitrary numerical values $a$ and $b$. The covariance of $X$ is obtained as follows
\begin{equation}
\text{Cov}[X] = \mathbb{E}[X - \mathbb{E}[X]]^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2
\end{equation}
The covariance measures the squared expected deviation from the mean. As stated above, the mean of a multivariate normal distribution $N(x; \mu, \Sigma)$ is $\mu$, and its covariance is $\Sigma$.

**Entropy**

A final concept of importance in this book is entropy. The entropy of a probability distribution is given by the following expression:
\begin{equation}
H_p(x) = \mathbb{E}[- \log_2 \Pr(x)]
\end{equation}
which resolves to
\begin{equation}
H_p(x) = - \sum_x p(x) \log_2 p(x) \quad \text{(discrete)}
\end{equation}
\begin{equation}
H_p(x) = - \int p(x) \log_2 p(x) \, dx \quad \text{(continuous)}
\end{equation}
The concept of entropy originates in information theory. The entropy is the expected information that the value of $x$ carries. In the discrete case, $- \log_2 \Pr(x)$ is the number of bits required to encode $x$ using an optimal encoding, assuming that $p(x)$ is the probability of observing $x$. In this book, entropy will be used in robotic information gathering, so as to express the information a robot may receive upon executing specific actions.

### 2.3 Robot Environment Interaction

Figure 2.1 illustrates the interaction of a robot with its environment. The environment, or world, 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. The effect of doing so is often somewhat unpredictable. Hence, each control action affects both the environment state, and the robot’s internal belief with regards to this state.

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 the state as the collection of all aspects of the robot and its environment that can impact the future. Certain state variables tend to change over time, such as the whereabouts of people in the vicinity of a robot. Others tend to remain static, such as the location of walls in (most) buildings. State that changes will be called dynamic state, which distinguishes it from static state, 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 $x_t$; 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 (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).

- In robot manipulation, the pose includes variables for the configuration of the robot's actuators. For example, they might include the joint angles of revolute joints. 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 configuration is often referred to as kinematic state.

- The robot velocity and the velocities of its joints are commonly referred to as dynamic state. A rigid robot moving through space is characterized by up to six velocity variables, one for each pose variable. Dynamic state will play only a minor role in this book.

- The location and features of surrounding objects in the environment are also state variables. 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

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. Some of 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 \( t = 0, 1, 2 \ldots \). 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 separate them throughout this book. The interaction is illustrated in Figure 2.1.

- **Environment 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 collection as the data (regardless of whether they are being memorized or not). In accordance with the two types of environment interactions, the robot has access to two different data streams.

- **Environment 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 laser 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 \( s_t \).

- **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, control conveys information regarding 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 state. Even though odometers are sensors, we will treat odometry as control data, since they measure the effect of a control action.
  
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 \( t_1 \leq t_2 \).

\[
\begin{align*}
\text{Measurement:} & \quad s_{t_1:t_2} = s_{t_1}, s_{t_1+1}, s_{t_1+2}, \ldots, s_{t_2} \\
\text{Control:} & \quad u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}
\end{align*}
\]

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. We therefore assume that there is exactly one control data item per time step \( t \), and include as legal action “do-nothing”.

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. Environment 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. By no means is our distinction intended to suggest that actions and perceptions are separated in time. Rather, perception and control takes place concurrently. Our 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 $x_t$ is generated stochastically from the state $x_{t-1}$. 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.

$P(x_t | x_{t-1}, z_{t-1}, u_{t-1})$

Notice that through no particular motivation we assume here that the robot executes a control action $u_t$ first, and then takes a measurement $z_t$.

An important insight is the following: 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 in time, that is, $u_{t-1}$ and $z_{t-1}$. 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:

$$P(x_t | x_{t-1}, z_{t-1}, u_{t-1}) \equiv P(x_t | x_{t-1}, u_t)$$

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. It is the primary reason why many of the algorithms presented in the book are computationally tractable.

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

$$P(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) \equiv P(z_t | x_t)$$

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.

---

**Figure 2.2** The dynamic Bayes network that characterizes the evolution of controls, states, and measurements.

![Dynamic Bayes Network](image)

This discussion leaves open as to what the two resulting conditional probabilities are: $P(x_t | x_{t-1}, u_t)$ and $P(x_t | z_t)$. The probability $P(x_t | x_{t-1}, u_t)$ 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 $P(x_t | x_{t-1}, u_t)$ 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 $P(x_t' | u_t, x_t)$, where $x'$ is the successor and $x$ the predecessor state.

The probability $P(x_t | z_t)$ is called the measurement probability. It also may not depend on the time index $t$, in which case it shall be written as $P(z_t | x_t)$. The measurement probability specifies the probabilistic law according to which measurements $z$ are generated from the environmental state $x$. It is appropriate to think of measurements as 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 2.2 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).

### 2.3.4 Belief Distributions

**BELIEF**

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 $x_t = (14.12, 12.7, 45^\circ)$ in some global coordinate sys-
INFORMATION STATE

Bayes Filters

Belief is the probability distribution over the state at time $t$, conditioned on all past measurements $z_{1:t-1}$ and all past controls $u_{1:t}$. The basic Bayes filter is recursive, that is, the belief $\text{bel}(x_t)$ at time $t$ is calculated from

\begin{align}
\text{bel}(x_t) &= P(x_t | z_{1:t}, u_{1:t}) \\
&= \text{bel}(x_t) \times \frac{P(z_t | x_t) \text{bel}(x_t)}{P(z_t | u_{1:t})}
\end{align}

Bayesian filtering is generally known as probability propagation. However, in the context of probabilistic filtering, this terminology reflects the fact that the measurement at time $t$ is obtained by the integral (sum) of the product of two distributions: the prior assigned to $x_t$ 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 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 $\text{bel}(x_t)$ 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.

Table 2.1 The general algorithm for Bayes filtering.

| Algorithm Bayes_filter(\text{bel}(x_{t-1}), u_t, z_t): |
| for all $x_t$ do |
| $\text{bel}(x_t) = \int P(x_t | u_t, x_{t-1}) \text{bel}(x_{t-1}) \, dx_{t-1}$ |
| $\text{bel}(x_t) = \eta P(x_t | x_{t-1}) \text{bel}(x_{t-1})$ |
| endfor |
| return $\text{bel}(x_t)$ |

the belief $\text{bel}(x_{t-1})$ at time $t - 1$. Its input is the belief $\text{bel}$ at time $t - 1$, along with the most recent control $u_t$ and the most recent measurement $z_t$. Its output is the belief $\text{bel}(x_t)$ at time $t$. Table 2.1 only depicts a single iteration of the Bayes Filter algorithm: the update rule. This update rule is applied recursively, to calculate the belief $\text{bel}(x_t)$ from the belief $\text{bel}(x_{t-1})$, 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 $\text{bel}(x_t)$ 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.
assigns zero probability anywhere else. If one is entirely ignorant about the initial value $x_0$, $\text{bel}(x_0)$ may be initialized using a uniform distribution over the domain of $x_0$ (or a related distribution from the Dirichlet family of distributions). Partial knowledge of the initial value $x_0$ can be 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.

### Example

Our illustration of the Bayes filter algorithm is based on the scenario in Figure 2.3, which shows a robot estimating the state of a door. 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:

- $\text{bel}(X_0 = \text{open}) = 0.5$
- $\text{bel}(X_0 = \text{closed}) = 0.5$

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

- $p(Z_t = \text{sense_open} \mid X_t = \text{is_open}) = 0.6$
- $p(Z_t = \text{sense_closed} \mid X_t = \text{is_open}) = 0.4$
- $p(Z_t = \text{sense_open} \mid X_t = \text{is_closed}) = 0.2$
- $p(Z_t = \text{sense_closed} \mid X_t = \text{is_closed}) = 0.8$

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 an erroneous 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. If it is closed, the robot has a 0.8 chance that it will open afterwards.

- $p(X_{t+1} = \text{is_open} \mid U_t = \text{push}, X_t = \text{is_open}) = 1$
- $p(X_{t+1} = \text{is_closed} \mid U_t = \text{push}, X_t = \text{is_open}) = 0$
- $p(X_{t+1} = \text{is_open} \mid U_t = \text{push}, X_t = \text{is_closed}) = 0.8$
- $p(X_{t+1} = \text{is_closed} \mid U_t = \text{push}, X_t = \text{is_closed}) = 0.2$

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:

- $p(X_{t+1} = \text{is_open} \mid U_t = \text{do_nothing}, X_t = \text{is_open}) = 1$
- $p(X_{t+1} = \text{is_closed} \mid U_t = \text{do_nothing}, X_t = \text{is_open}) = 0$
- $p(X_{t+1} = \text{is_open} \mid U_t = \text{do_nothing}, X_t = \text{is_closed}) = 0$
- $p(X_{t+1} = \text{is_closed} \mid U_t = \text{do_nothing}, X_t = \text{is_closed}) = 1$

Suppose at time $t = 1$, 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 $\text{bel}(X_0)$, the control $u_1 = \text{do_nothing}$, and the measurement $\text{sense_open}$ as input. Since the state space is finite, the integral in line 3 turns into a finite sum:

$$
\text{bel}(x_1) = \int p(x_1 \mid u_1, x_0) \text{bel}(x_0) \, dx_0
$$

$$
= \sum_{x_0} p(x_1 \mid u_1, x_0) \text{bel}(x_0)
$$

$$
= p(x_1 \mid U_1 = \text{do_nothing}, X_0 = \text{is_open}) \text{bel}(X_0 = \text{is_open}) + p(x_1 \mid U_1 = \text{do_nothing}, X_0 = \text{is_closed}) \text{bel}(X_0 = \text{is_closed})
$$
Recursive State Estimation

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

$$\text{bel}(X_1 = \text{is\_open}) = p(X_1 = \text{is\_open} | U_1 = \text{do\_nothing}, X_0 = \text{is\_open})$$

$$\text{bel}(X_0 = \text{is\_open}) + p(X_1 = \text{is\_open} | U_1 = \text{do\_nothing}, X_0 = \text{is\_closed})$$

$$\text{bel}(X_0 = \text{is\_closed}) = 1.05 + 0.5 = 0.5$$

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

$$\text{bel}(X_1 = \text{is\_closed}) = p(X_1 = \text{is\_closed} | U_1 = \text{do\_nothing}, X_0 = \text{is\_open})$$

$$\text{bel}(X_0 = \text{is\_open}) + p(X_1 = \text{is\_closed} | U_1 = \text{do\_nothing}, X_0 = \text{is\_closed})$$

$$\text{bel}(X_0 = \text{is\_closed}) = 0.05 + 1.05 = 0.5$$

The fact that the belief $\text{bel}(x_1)$ equals our prior belief $\text{bel}(x_0)$ should not surprise, as the action 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

$$\text{bel}(x_1) = \eta p(Z_t = \text{sense\_open} | x_1) \text{bel}(x_1)$$

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

$$\text{bel}(X_1 = \text{is\_open}) = \eta p(Z_t = \text{sense\_open} | X_1 = \text{is\_open}) \text{bel}(X_1 = \text{is\_open})$$

$$\text{bel}(X_1 = \text{is\_closed}) = \eta 0.6 - 0.5 = \eta 0.3$$

and

$$\text{bel}(X_1 = \text{is\_closed}) = \eta p(Z_t = \text{sense\_open} | X_1 = \text{is\_closed}) \text{bel}(X_1 = \text{is\_closed})$$

$$\text{bel}(X_1 = \text{is\_open}) = \eta 0.2 - 0.5 = \eta 0.1$$

The normalizer $\eta$ is now easily calculated:

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

Hence, we have

$$\text{bel}(X_1 = \text{is\_open}) = 0.75$$

$$\text{bel}(X_1 = \text{is\_closed}) = 0.25$$

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

$$\text{bel}(X_2 = \text{is\_open}) = 1.075 + 0.8 - 0.25 = 0.95$$

$$\text{bel}(X_2 = \text{is\_closed}) = 0.075 + 0.2 - 0.25 = 0.05$$

and

$$\text{bel}(X_2 = \text{is\_open}) = \eta 0.6 - 0.95 \approx 0.983$$

$$\text{bel}(X_2 = \text{is\_closed}) = \eta 0.2 - 0.05 \approx 0.017$$

At this point, the robot believes that with 0.983 probability the door is open.

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 $p(x_t | z_1:t, u_1:t)$ from the corresponding posterior one time step earlier, $p(x_{t-1} | z_1:t-1, u_1:t-1)$. The correctness follows then by induction under the assumption that we correctly initialized the prior belief $\text{bel}(x_0)$ at time $t = 0$.

Our derivation requires that the state $x_t$ is complete, as defined in Chapter 2.3.1, and it requires that controls are chosen at random. The first step of our derivation involves the application of Bayes rule (2.16) to the target posterior:

$$p(x_t | z_1:t, u_1:t) = \frac{p(z_t | x_t, z_1:t-1, u_1:t) p(x_t | z_1:t-1, u_1:t)}{p(z_t | z_1:t-1, u_1:t)}$$

$$= \eta p(z_t | x_t, z_1:t-1, u_1:t) p(x_t | z_1:t-1, u_1:t)$$

We now exploit the assumption that our state is complete. In Chapter 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:

\[
p(x_t \mid x_{t-1}, u_{t-1}, u_{t-1}) = p(x_t \mid x_{t-1})
\]

Such a statement is another example of conditional independence. It allows us to simplify (2.35) as follows:

\[
p(x_t \mid z_{t-1}, u_{t-1}) = \eta p(x_t \mid x_{t-1}) p(x_{t-1} \mid z_{t-1-1}, u_{t-1})
\]

and hence

\[
\text{bel}(x_t) = \eta p(x_t \mid x_{t-1}) \text{bel}(x_{t-1})
\]

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

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

\[
\text{bel}(x_t) = p(x_0 | z_{t-1-1}, u_{t-1}) \int p(x_t | x_{t-1-1}, z_{t-1-1}, u_{t-1}) p(x_{t-1} | z_{t-1-1}, u_{t-1}) d x_{t-1}
\]

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

\[
p(x_t | z_{t-1-1}, u_{t-1}) = p(x_t | x_{t-1}, u_t)
\]

Here we retain the control variable \(u_t\), since it does not predetermine the state \(x_{t-1}\). In fact, the reader should quickly convince herself that \(p(x_t | x_{t-1-1}, u_t) \neq p(x_t | x_{t-1})\).

Finally, we note that the control \(u_t\) can safely be omitted from the set of conditioning variables in \(p(x_{t-1} | x_{t-1-1}, u_{t-1})\) for randomly chosen controls. This gives us the recursive update equation

\[
\text{bel}(x_t) = \int p(x_t | x_{t-1}, u_t) p(x_{t-1} | z_{t-1-1}, u_{t-1}) d x_{t-1}
\]

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.

### 2.4 Bayes Filters

Any concrete implementation of this algorithm requires three probability distributions: The initial belief \(p(x_0)\), the measurement probability \(p(z_t | x_t)\), and the state transition probability \(p(x_t | x_{t-1})\). We have not yet specified these densities for actual robot systems. But we will soon: Chapter 5 is entirely dedicated to \(p(x_t | u_t, x_{t-1})\) and Chapter 6 to \(p(z_t | x_t)\). We also need a representation for the belief \(\text{bel}(x_t)\), which will be discussed in Chapters 3 and 4.

#### 2.4.4 The Markov Assumption

**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 \(p(x_t | x_{t-1})\) and \(p(x_t | u_t, x_{t-1})\) (e.g., an error in the map for a localizing robot),
- 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 controls (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, however, one should 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. These assumptions then give rise to different types of posterior distributions, and the algorithms for computing them 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 \( p(z_t | x_t) \) and the state transition probability \( p(x_t | u_t, x_{t-1}) \). 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 and the robot.

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

- 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, possibly as the effect of robot controls. 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 according to which 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.