PROBABILISTIC ROBOTICS

Sebastian THRUN
Stanford University
Stanford, CA

■

Wolfram BURGARD
University of Freiburg
Freiburg, Germany

■

Dieter FOX
University of Washington
Seattle, WA

CONTENTS

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 fi ltering.

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.

2.4.3 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):

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

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.

3 GAUSSIAN FILTERS

3.1 INTRODUCTION

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

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

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

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

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

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

This chapter introduces the two basic Gaussian filter algorithms.

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

3.2 THE KALMAN FILTER

3.2.1 Linear Gaussian Systems

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

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

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

$$x_t \quad = \quad A_t x_{t-1} + B_t u_t + \varepsilon_t \ . \qquad (3.2)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 The Kalman Filter Algorithm

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

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

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

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

3.2.3 Illustration

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

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

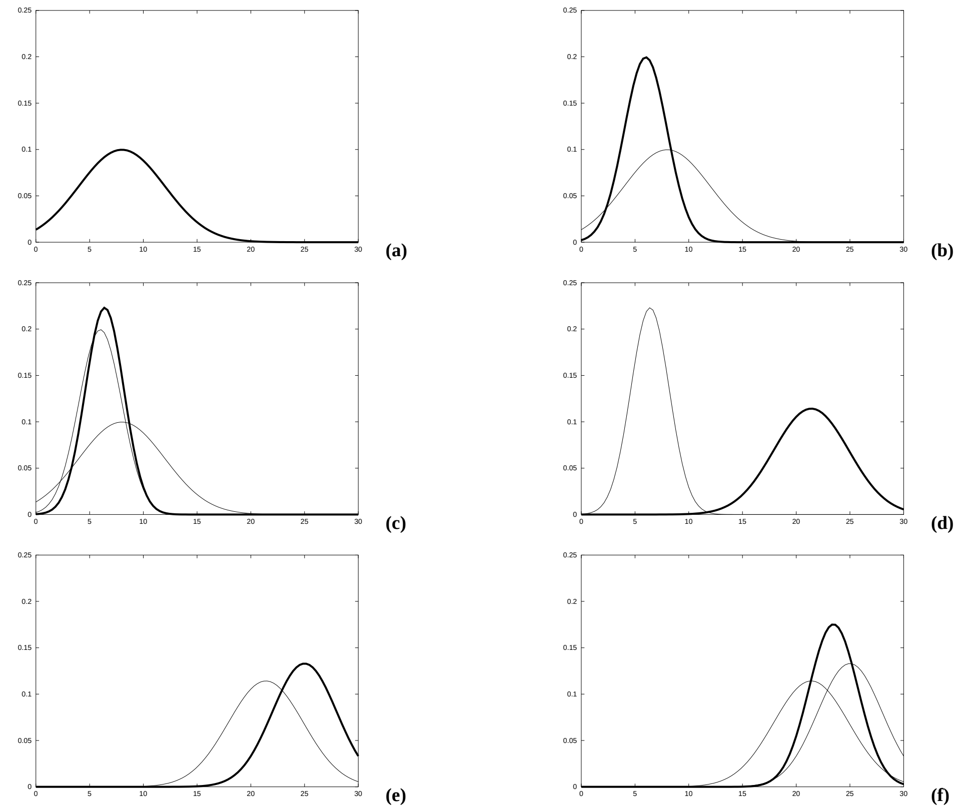

Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement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

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

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