6 Robot Perception

6.1 Introduction

Environment measurement models comprise the second domain-specific model in probabilistic robotics, next to motion models. Measurement models describe the formation process by which sensor measurements are generated in the physical world. Today's robots use a variety of different sensor modalities, such as tactile sensors, range sensors, or cameras. The specifics of the model depends on the sensor: Imaging sensors are best modeled by projective geometry, whereas sonar sensors are best modeled by describing the sound wave and its reflection on surfaces in the environment.

Probabilistic robotics explicitly models the noise in sensor measurements. Such models account for the inherent uncertainty in the robot's sensors. Formally, the measurement model is defined as a conditional probability distribution $p(z_t \mid x_t, m)$, where x_t is the robot pose, z_t is the measurement at time t, and m is the map of the environment. Although we mainly address range-sensors throughout this chapter, the underlying principles and equations are not limited to this type of sensors. Instead the basic principle can be applied to any kind of sensor, such as a camera or a bar-code operated landmark detector.

SONAR RANGE SCAN

To illustrate the basic problem of mobile robots that use their sensors to perceive their environment, Figure 6.1a shows a typical *sonar range scan* obtained in a corridor with a mobile robot equipped with a cyclic array of 24 ultrasound sensors. The distances measured by the individual sensors are depicted in light gray and the map of the environment is shown in black. Most of these measurements correspond to the distance of the nearest object in the measurement cone; some measurements, however, have failed to detect any object.

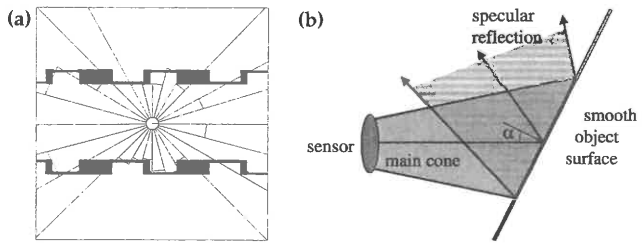


Figure 6.1 (a) Typical ultrasound scan of a robot in its environment. (b) A misreading in ultrasonic sensing. This effect occurs when firing a sonar signal towards a reflective surface at an angle α that exceeds half the opening angle of the sensor.

The inability for sonar to reliably measure range to nearby objects is often paraphrased as sensor noise. Technically, this noise is quite predictable: When measuring smooth surfaces (such as walls), the reflection is usually *specular*, and the wall effectively becomes a mirror for the sound wave. This can be problematic when hitting a smooth surface at an angle. Here the echo may travel into a direction other than the sonar sensor, as illustrated in Figure 6.1b. This effect often leads to overly large range measurements when compared to the true distance to the nearest object in the main cone. The likelihood of this to happen depends on a number of properties, such as the surface material, the angle between the surface normal and the direction of the sensor cone, the range of the surface, the width of the main sensor cone, and the sensitivity of the sonar sensor. Other errors, such as short readings, may be caused by cross-talk between different sensors (sound is slow!) or by unmodeled objects in the proximity of the robot, such as people.

SPECULAR REFLECTION

LASER RANGE SCAN

Figure 6.2 shows a typical *laser range scan*, acquired with a 2-D laser range finder. Laser is similar to sonar in that it also actively emits a signal and records its echo, but in the case of laser the signal is a light beam. A key difference to sonars is that lasers provide much more focused beams. The specific laser in Figure 6.2 is based on a time-of-flight measurement, and measurements are spaced in one degree increments.

As a rule of thumb, the more accurate a sensor model, the better the results—though there are some important caveats that were already discussed in Chapter 2.4.4. In practice, however, it is often impossible to model

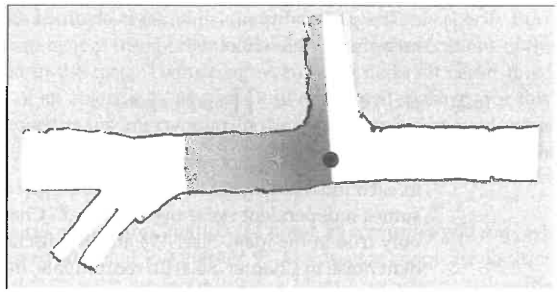


Figure 6.2 A typical laser range scan, acquired with a SICK LMS laser. The environment shown here is a coal mine. Image courtesy of Dirk Hähnel, University of Freiburg.

a sensor accurately, primarily due to the complexity of physical phenomena.

Often, the response characteristics of a sensor depends on variables we prefer not to make explicit in a probabilistic robotics algorithm (such as the surface material of walls, which for no particular reason is commonly not considered in robotic mapping). Probabilistic robotics accommodates inaccuracies of sensor models in the stochastic aspects: By modeling the measurement process as a conditional probability density, $p(z_t \mid x_t)$, instead of a deterministic function $z_t = f(x_t)$, the uncertainty in the sensor model can be accommodated in the non-deterministic aspects of the model. Herein lies a key advantage of probabilistic techniques over classical robotics: in practice, we can get away with extremely crude models. However, when devising a probabilistic model, care has to be taken to capture the different types of uncertainties that may affect a sensor measurement.

Many sensors generate more than one numerical measurement value when queried. For example, cameras generate entire arrays of values (brightness, saturation, color); similarly, range finders usually generate entire scans of ranges. We will denote the number of such measurement values within a measurement z_t by K, hence we can write:

$$z_t = \{z_t^1, \ldots, z_t^K\} \tag{6.1}$$

We will use z_t^k to refer to an individual measurement (e.g., one range value).

The probability $p(z_t \mid x_t, m)$ is obtained as the product of the individual measurement likelihoods

$$p(z_t \mid x_t, m) = \prod_{k=1}^{K} p(z_t^k \mid x_t, m) \tag{6.2}$$

Technically, this amounts to an *independence assumption* between the noise in each individual measurement beam—just as our Markov assumption assumes independent noise over time (c.f., Chapter 2.4.4). This assumption is only true in the ideal case. We already discussed possible causes of dependent noise in Chapter 2.4.4. To recapitulate, dependencies typically exist due to a range of factors: people, who often corrupt measurements of several adjacent sensors; errors in the model m; approximations in the posterior; and so on. For now, however, we will simply not worry about violations of the independence assumption, as we will return to this issue in later chapters.

6.2 Maps

To express the process of generating measurements, we need to specify the environment in which a measurement is generated. A *map* of the environment is a list of objects in the environment and their locations. We have already informally discussed maps in the previous chapter, where we developed robot motion models that took into consideration the occupancy of different locations in the world. Formally, a map m is a list of objects in the environment along with their properties:

$$m = \{m_1, m_2, \ldots, m_N\} \tag{6.3}$$

Here N is the total number of objects in the environment, and each m_n with $1 \leq n \leq N$ specifies a property. Maps are usually indexed in one of two ways, known as *feature-based* and *location-based*. In feature-based maps, n is a feature index. The value of m_n contains, next to the properties of a feature, the Cartesian location of the feature. In location-based maps, the index n corresponds to a specific location. In planar maps, it is common to denote a map element by $m_{x,y}$ instead of m_n, to make explicit that $m_{x,y}$ is the property of a specific world coordinate, $(x \ y)$.

VOLUMETRIC MAPS

Both types of maps have advantages and disadvantages. Location-based maps are *volumetric*, in that they offer a label for any location in the world. Volumetric maps contain information not only about objects in the environment, but also about the absence of objects (e.g., free-space). This is quite

different in feature-based maps. *Feature-based maps* only specify the shape of the environment at the specific locations, namely the locations of the objects contained in the map. Feature representation makes it easier to adjust the position of an object; e.g., as a result of additional sensing. For this reason, feature-based maps are popular in the robotic mapping field, where maps are constructed from sensor data. In this book, we will encounter both types of maps—in fact, we will occasionally move from one representation to the other.

A classical map representation is known as *occupancy grid map*, which will be discussed in detail in Chapter 9. Occupancy maps are location-based: They assign to each x-y coordinate a binary occupancy value that specifies whether or not a location is occupied with an object. Occupancy grid maps are great for mobile robot navigation: They make it easy to find paths through the unoccupied space.

Throughout this book, we will drop the distinction between the physical world and the map. Technically, sensor measurements are caused by physical objects, not the map of those objects. However, it is tradition to condition sensor models on the map m; hence we will adopt a notation that suggests measurements depend on the map.

6.3 Beam Models of Range Finders

Range finders are among the most popular sensors in robotics. Our first *measurement model* in this chapter is therefore an approximative physical model of range finders. Range finders measure the range to nearby objects. Range may be measured along a beam—which is a good model of the workings of laser range finders—or within a cone—which is the preferable model of ultrasonic sensors.

6.3.1 The Basic Measurement Algorithm

Our model incorporates four types of measurement errors, all of which are essential to making this model work: small measurement noise, errors due to unexpected objects, errors due to failures to detect objects, and random unexplained noise. The desired model $p(z_t \mid x_t, m)$ is therefore a mixture of four densities, each of which corresponds to a particular type of error:

1. **Correct range with local measurement noise.** In an ideal world, a range finder would always measure the correct range to the nearest object in its

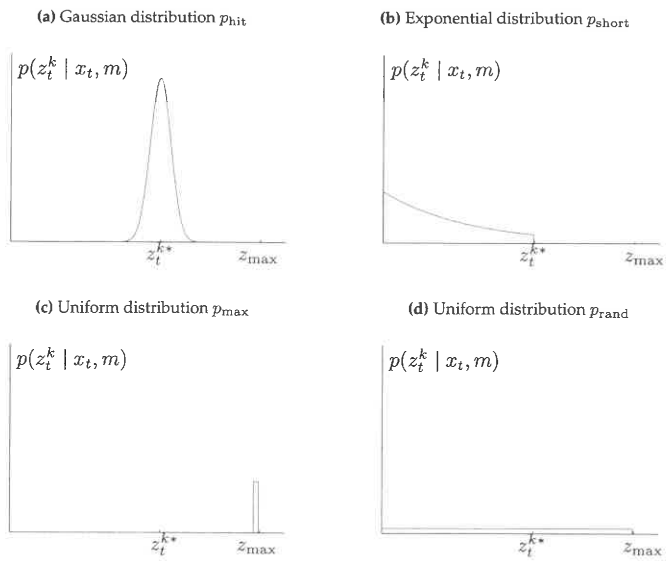


Figure 6.3 Components of the range finder sensor model. In each diagram the horizontal axis corresponds to the measurement z_t^k, the vertical to the likelihood.

measurement field. Let us use z_t^{k*} to denote the "true" range of the object measured by z_t^k. In location-based maps, the range z_t^{k*} can be determined using *ray casting*; in feature-based maps, it is usually obtained by searching for the closest feature within a measurement cone. However, even if the sensor correctly measures the range to the nearest object, the value it returns is subject to error. This error arises from the limited resolution of range sensors, atmospheric effect on the measurement signal, and so on. This *measurement noise* is usually modeled by a narrow Gaussian with mean z_t^{k*} and standard deviation σ_{hit}. We will denote the Gaussian by p_{hit}. Figure 6.3a illustrates this density p_{hit}, for a specific value of z_t^{k*}.

MEASUREMENT NOISE

In practice, the values measured by the range sensor are limited to the interval $[0; z_{\text{max}}]$, where z_{max} denotes the maximum sensor range. Thus,

the measurement probability is given by

$$p_{\text{hit}}(z_t^k \mid x_t, m) = \begin{cases} \eta\, \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \le z_t^k \le z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

where z_t^{k*} is calculated from x_t and m via ray casting, and $\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2)$ denotes the univariate normal distribution with mean z_t^{k*} and standard deviation σ_{hit}:

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) = \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}}\, e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{\text{hit}}^2}} \tag{6.5}$$

The normalizer η evaluates to

$$\eta = \left(\int_0^{z_{\text{max}}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2)\, dz_t^k \right)^{-1} \tag{6.6}$$

The standard deviation σ_{hit} is an intrinsic noise parameter of the measurement model. Below we will discuss strategies for setting this parameter.

2. **Unexpected objects.** Environments of mobile robots are dynamic, whereas maps m are static. As a result, objects not contained in the map can cause range finders to produce surprisingly short ranges—at least when compared to the map. A typical example of moving objects are people that share the operational space of the robot. One way to deal with such objects is to treat them as part of the state vector and estimate their location; another, much simpler approach, is to treat them as sensor noise. Treated as sensor noise, unmodeled objects have the property that they cause ranges to be shorter than z_t^{k*}, not longer.

   The likelihood of sensing unexpected objects decreases with range. To see, imagine there are two people that independently and with the same fixed likelihood show up in the perceptual field of a proximity sensor. One person's range is r_1, and the second person's range is r_2. Let us further assume that $r_1 < r_2$, without loss of generality. Then we are more likely to measure r_1 than r_2. Whenever the first person is present, our sensor measures r_1. However, for it to measure r_2, the second person must be present *and* the first must be absent.

   Mathematically, the probability of range measurements in such situations is described by an *exponential distribution*. The parameter of this distribution, λ_{short}, is an intrinsic parameter of the measurement model. According to the definition of an exponential distribution we obtain the following

equation for $p_{\text{short}}(z_t^k \mid x_t, m)$:

$$p_{\text{short}}(z_t^k \mid x_t, m) = \begin{cases} \eta\, \lambda_{\text{short}}\, e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \le z_t^k \le z_t^{k*} \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

As in the previous case, we need a normalizer η since our exponential is limited to the interval $[0; z_t^{k*}]$. Because the cumulative probability in this interval is given as

$$\begin{aligned} \int_0^{z_t^{k*}} \lambda_{\text{short}}\, e^{-\lambda_{\text{short}} z_t^k}\, dz_t^k &= -e^{-\lambda_{\text{short}} z_t^{k*}} + e^{-\lambda_{\text{short}} 0} \\ &= 1 - e^{-\lambda_{\text{short}} z_t^{k*}} \end{aligned} \tag{6.8}$$

the value of η can be derived as:

$$\eta = \frac{1}{1 - e^{-\lambda_{\text{short}} z_t^{k*}}} \tag{6.9}$$

Figure 6.3b depicts this density graphically. This density falls off exponentially with the range z_t^k.

3. **Failures.** Sometimes, obstacles are missed altogether. For example, this happens frequently for sonar sensors as a result of specular reflections. Failures also occur with laser range finders when sensing black, light-absorbing objects, or for some laser systems when measuring objects in bright sunlight. A typical result of a *sensor failure* is a *max-range measurement*: the sensor returns its maximum allowable value z_{max}. Since such events are quite frequent, it is necessary to explicitly model max-range measurements in the measurement model.

SENSOR FAILURE

We will model this case with a point-mass distribution centered at z_{max}:

$$p_{\text{max}}(z_t^k \mid x_t, m) = I(z = z_{\text{max}}) = \begin{cases} 1 & \text{if } z = z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

Here I denotes the indicator function that takes on the value 1 if its argument is true, and is 0 otherwise. Technically, p_{max} does not possess a probability density function. This is because p_{max} is a discrete distribution. However, this shall not worry us here, as our mathematical model of evaluating the probability of a sensor measurement is not affected by the non-existence of a density function. (In our diagrams, we simply draw p_{max} as a very narrow uniform distribution centered at z_{max}, so that we can pretend a density exists).

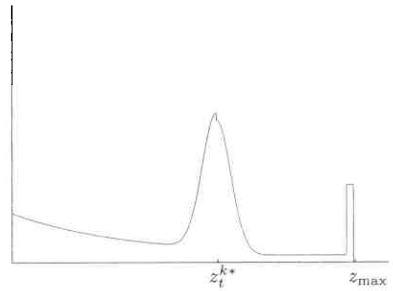


Figure 6.4 "Pseudo-density" of a typical mixture distribution $p(z_t^k \mid x_t, m)$.

4. **Random measurements.** Finally, range finders occasionally produce entirely *unexplainable measurements*. For example, sonars often generate phantom readings when they bounce off walls, or when they are subject to cross-talk between different sensors. To keep things simple, such measurements will be modeled using a uniform distribution spread over the entire sensor measurement range $[0; z_{\text{max}}]$:

UNEXPLAINABLE MEASUREMENTS

$$p_{\text{rand}}(z_t^k \mid x_t, m) = \begin{cases} \frac{1}{z_{\text{max}}} & \text{if } 0 \le z_t^k < z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

Figure 6.3d shows the density of the distribution p_{rand}.

These four different distributions are now mixed by a weighted average, defined by the parameters z_{hit}, z_{short}, z_{max}, and z_{rand} with $z_{\text{hit}} + z_{\text{short}} + z_{\text{max}} + z_{\text{rand}} = 1$.

$$p(z_t^k \mid x_t, m) = \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\text{max}} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k \mid x_t, m) \\ p_{\text{short}}(z_t^k \mid x_t, m) \\ p_{\text{max}}(z_t^k \mid x_t, m) \\ p_{\text{rand}}(z_t^k \mid x_t, m) \end{pmatrix} \tag{6.12}$$

A typical density resulting from this linear combination of the individual densities is shown in Figure 6.4 (with our visualization of the point-mass distribution p_{max} as a small uniform density). As the reader may notice, the basic characteristics of all four basic models are still present in this combined density.

1:	**Algorithm beam_range_finder_model**(z_t, x_t, m)**:**
2:	$q = 1$
3:	*for* $k = 1$ *to* K *do*
4:	*compute* z_t^{k*} *for the measurement* z_t^k *using ray casting*
5:	$p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m)$
6:	$\quad + z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$
7:	$q = q \cdot p$
8:	*return* q

Table 6.1 Algorithm for computing the likelihood of a range scan z_t, assuming conditional independence between the individual range measurements in the scan.

The range finder model is implemented by the algorithm **beam_range_finder_model** in Table 6.1. The input of this algorithm is a complete range scan z_t, a robot pose x_t, and a map m. Its outer loop (lines 2 and 7) multiplies the likelihood of individual sensor beams z_t^k, following Equation (6.2). Line 4 applies ray casting to compute the noise-free range for a particular sensor measurement. The likelihood of each individual range measurement z_t^k is computed in line 5, which implements the mixing rule for densities stated in (6.12). After iterating through all sensor measurements z_t^k in z_t, the algorithm returns the desired probability $p(z_t \mid x_t, m)$.

6.3.2 Adjusting the Intrinsic Model Parameters

In our discussion so far we have not addressed the question of how to choose the various parameters of the sensor model. These parameters include the mixing parameters z_{hit}, z_{short}, z_{max}, and z_{rand}. They also include the parameters σ_{hit} and λ_{short}. We will refer to the set of all intrinsic parameters as Θ. Clearly, the likelihood of any sensor measurement is a function of Θ. Thus, we will now discuss an algorithm for *adjusting model parameters*.

One way to determine the intrinsic parameters is to rely on data. Figure 6.5 depicts two series of 10,000 measurements obtained with a mobile robot traveling through a typical office environment. Both plots show only range measurements for which the expected range was approximately 3 meters (between 2.9m and 3.1m). The left plot depicts the data for sonar sensors, and the right plot the corresponding data for laser sensors. In both plots, the x-axis shows the number of the reading (from 1 to 10,000), and the y-axis is the range measured by the sensor.

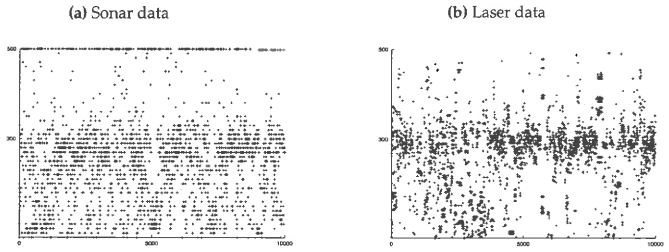


Figure 6.5 Typical data obtained with (a) a sonar sensor and (b) a laser-range sensor in an office environment for a "true" range of 300 cm and a maximum range of 500 cm.

Whereas most of the measurements are close to the correct range for both sensors, the behaviors of the sensors differ substantially. The ultrasound sensor appears to suffer from many more measurement noise and detection errors. Quite frequently it fails to detect an obstacle, and instead reports maximum range. In contrast, the laser range finder is more accurate. However, it also occasionally reports false ranges.

A perfectly acceptable way to set the intrinsic parameters Θ is by hand: simply eyeball the resulting density until it agrees with your experience. Another, more principled way is to learn these parameters from actual data. This is achieved by maximizing the likelihood of a reference data set $Z = \{z_i\}$ with associated positions $X = \{x_i\}$ and map m, where each z_i is an actual measurement, x_i is the pose at which the measurement was taken, and m is the map. The likelihood of the data Z is given by

$$p(Z \mid X, m, \Theta) \tag{6.13}$$

Our goal is to identify intrinsic parameters Θ that maximize this likelihood. Any estimator, or algorithm, that maximizes the likelihood of data is known as a *maximum likelihood estimator*, or *ML estimator*.

MAXIMUM LIKELIHOOD ESTIMATOR

Table 6.2 depicts the algorithm **learn_intrinsic_parameters**, which is an algorithm for calculating the maximum likelihood estimate for the intrinsic

1: **Algorithm learn_intrinsic_parameters**(Z, X, m):

2: *repeat until convergence criterion satisfied*

3: *for all* z_i *in* Z *do*

4: $\eta = [\, p_{\text{hit}}(z_i \mid x_i, m) + p_{\text{short}}(z_i \mid x_i, m) + p_{\text{max}}(z_i \mid x_i, m) + p_{\text{rand}}(z_i \mid x_i, m)\,]^{-1}$

5: *calculate* z_i^*

6: $e_{i,\text{hit}} = \eta\, p_{\text{hit}}(z_i \mid x_i, m)$

7: $e_{i,\text{short}} = \eta\, p_{\text{short}}(z_i \mid x_i, m)$

8: $e_{i,\text{max}} = \eta\, p_{\text{max}}(z_i \mid x_i, m)$

9: $e_{i,\text{rand}} = \eta\, p_{\text{rand}}(z_i \mid x_i, m)$

10: $z_{\text{hit}} = |Z|^{-1} \sum_i e_{i,\text{hit}}$

11: $z_{\text{short}} = |Z|^{-1} \sum_i e_{i,\text{short}}$

12: $z_{\text{max}} = |Z|^{-1} \sum_i e_{i,\text{max}}$

13: $z_{\text{rand}} = |Z|^{-1} \sum_i e_{i,\text{rand}}$

14: $\sigma_{\text{hit}} = \sqrt{\frac{1}{\sum_i e_{i,\text{hit}}} \sum_i e_{i,\text{hit}} (z_i - z_i^*)^2}$

15: $\lambda_{\text{short}} = \frac{\sum_i e_{i,\text{short}}}{\sum_i e_{i,\text{short}}\, z_i}$

16: *return* $\Theta = \{z_{\text{hit}}, z_{\text{short}}, z_{\text{max}}, z_{\text{rand}}, \sigma_{\text{hit}}, \lambda_{\text{short}}\}$

Table 6.2 Algorithm for learning the intrinsic parameters of the beam-based sensor model from data.

parameters. As we shall see below, the algorithm is an instance of the *expectation maximization* (EM) algorithm, an iterative procedure for estimating ML parameters.

Initially, the algorithm **learn_intrinsic_parameters** in Table 6.2 requires a good initialization of the intrinsic parameters σ_{hit} and λ_{short}. In lines 3 through 9, it estimates auxiliary variables: Each $e_{i,\text{xxx}}$ is the probability that the measurement z_i is caused by "xxx," where "xxx" is chosen from the four aspects of the sensor model, hit, short, max, and random. Subsequently, it estimates the intrinsic parameters in lines 10 through 15. The intrinsic parameters, however, are a function of the expectations calculated before. Adjusting the intrinsic parameters causes the expectations to change, for which reason the algorithm has to be iterated. However, in practice the iteration converges quickly, and a dozen iterations are usually sufficient to give good results.

Figure 6.6 graphically depicts four examples of data and the ML measurement model calculated by **learn_intrinsic_parameters**. The first row shows approximations to data recorded with the ultrasound sensor. The second row contains plots of two functions generated for laser range data. The columns correspond to different "true" ranges. The data is organized in histograms. One can clearly see the differences between the different graphs. The smaller the range z_t^{k*} the more accurate the measurement. For both sensors the Gaussians are narrower for the shorter range than they are for the longer measurement. Furthermore, the laser range finder is more accurate than the ultrasound sensor, as indicated by the narrower Gaussians and the smaller number of maximum range measurements. The other important thing to notice is the relatively high likelihood of short and random measurements. This large error likelihood has a disadvantage and an advantage: On the negative side, it reduces the information in each sensor reading, since the difference in likelihood between a hit and a random measurement is small. On the positive side this model is less susceptible to unmodeled *systematic* perturbations, such as people who block the robot's path for long periods of time.

Figure 6.7 illustrates the learned sensor model in action. Shown in Figure 6.7a is a 180 degree range scan. The robot is placed in a previously acquired occupancy grid map at its true pose. Figure 6.7b plots a map of the environment along with the likelihood $p(z_t \mid x_t, m)$ of this range scan projected into x-y-space (by maximizing over the orientation θ). The darker a location, the more likely it is. As is easily seen, all regions with high likelihood are located in the corridor. This comes at little surprise, as the specific scan is geometrically more consistent with corridor locations than with locations inside any of the rooms. The fact that the probability mass is spread out throughout the corridor suggests that a single sensor scan is insufficient to determine the robot's exact pose. This is largely due to the symmetry of the corridor. The fact that the posterior is organized in two narrow horizontal bands is due to the fact that the orientation of the robot is unknown: each of these bands corresponds to one of the two surviving heading directions of the robot.

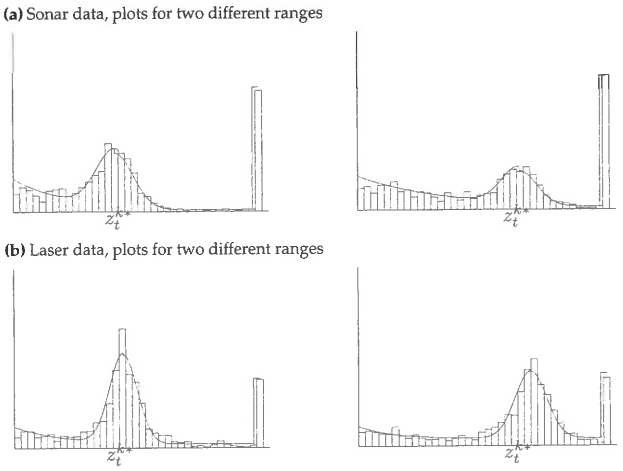


Figure 6.6 Approximation of the beam model based on (a) sonar data and (b) laser range data. The sensor models depicted on the left were obtained by a maximum likelihood approximation to the data sets depicted in Figure 6.5.

6.3.3 Mathematical Derivation of the Beam Model

To derive the ML estimator, it shall prove useful to introduce auxiliary variables c_i, the so-called correspondence variable. Each c_i can take on one of four values, hit, short, max, and random, corresponding to the four possible mechanisms that might have produced a measurement z_i.

Let us first consider the case in which the c_i's are known. We know which of the four mechanisms described above caused each measurement z_i. Based on the values of the c_i's, we can decompose Z into four disjoint sets, Z_{hit}, Z_{short}, Z_{max}, and Z_{rand}, which together comprise the set Z. The ML estimators for the intrinsic parameters z_{hit}, z_{short}, z_{max}, and z_{rand} are simply the

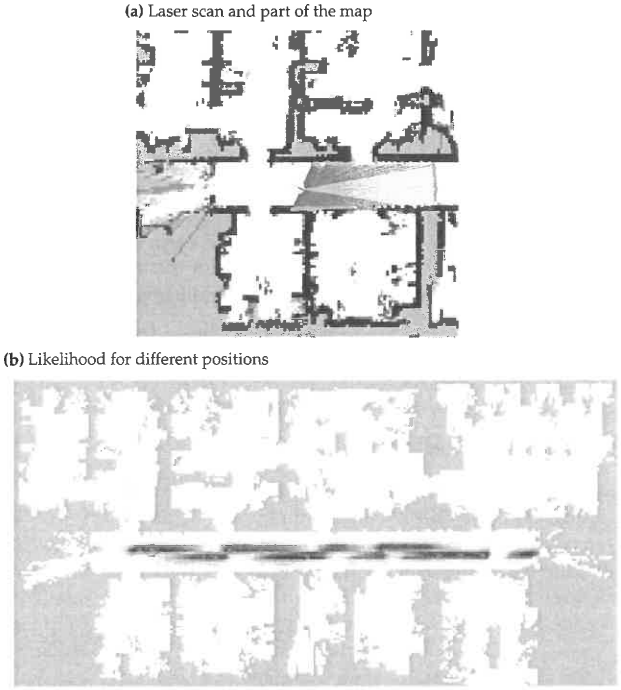


Figure 6.7 Probabilistic model of perception: (a) Laser range scan, projected into a previously acquired map m. (b) The likelihood $p(z_t \mid x_t, m)$, evaluated for all positions x_t and projected into the map (shown in gray). The darker a position, the larger $p(z_t \mid x_t, m)$.

normalized ratios:

$$\begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\text{max}} \\ z_{\text{rand}} \end{pmatrix} = |Z|^{-1} \begin{pmatrix} |Z_{\text{hit}}| \\ |Z_{\text{short}}| \\ |Z_{\text{max}}| \\ |Z_{\text{rand}}| \end{pmatrix} \tag{6.14}$$

The remaining intrinsic parameters, σ_{hit} and λ_{short}, are obtained as follows.

For the data set Z_{hit}, we get from (6.5)

$$p(Z_{\text{hit}} \mid X, m, \Theta) = \prod_{z_i \in Z_{\text{hit}}} p_{\text{hit}}(z_i \mid x_i, m, \Theta) \tag{6.15}$$

$$= \prod_{z_i \in Z_{\text{hit}}} \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}} e^{-\frac{1}{2}\frac{(z_i - z_i^*)^2}{\sigma_{\text{hit}}^2}}$$

Here z_i^* is the "true" range, computed from the pose x_i and the map m. A classic trick of ML estimation is to maximize the logarithm of the likelihood, instead of the likelihood directly. The logarithm is a strictly monotonic function, hence the maximum of the log-likelihood is also the maximum of the original likelihood. The log-likelihood is given by

$$\log p(Z_{\text{hit}} \mid X, m, \Theta) = \sum_{z_i \in Z_{\text{hit}}} \left[-\frac{1}{2} \log 2\pi\sigma_{\text{hit}}^2 - \frac{1}{2} \frac{(z_i - z_i^*)^2}{\sigma_{\text{hit}}^2} \right] \tag{6.16}$$

which is now easily transformed as follows

$$\begin{aligned} &\log p(Z_{\text{hit}} \mid X, m, \Theta) \\ &= -\frac{1}{2} \sum_{z_i \in Z_{\text{hit}}} \left[\log 2\pi\sigma_{\text{hit}}^2 + \frac{(z_i - z_i^*)^2}{\sigma_{\text{hit}}^2} \right] \\ &= -\frac{1}{2} \left[|Z_{\text{hit}}| \log 2\pi + 2|Z_{\text{hit}}| \log \sigma_{\text{hit}} + \sum_{z_i \in Z_{\text{hit}}} \frac{(z_i - z_i^*)^2}{\sigma_{\text{hit}}^2} \right] \\ &= \text{const.} - |Z_{\text{hit}}| \log \sigma_{\text{hit}} - \frac{1}{2\sigma_{\text{hit}}^2} \sum_{z_i \in Z_{\text{hit}}} (z_i - z_i^*)^2 \end{aligned} \tag{6.17}$$

The derivative of this expression in the intrinsic parameter σ_{hit} is as follows:

$$\frac{\partial \log p(Z_{\text{hit}} \mid X, m, \Theta)}{\partial \sigma_{\text{hit}}} = -\frac{|Z_{\text{hit}}|}{\sigma_{\text{hit}}} + \frac{1}{\sigma_{\text{hit}}^3} \sum_{z_i \in Z_{\text{hit}}} (z_i - z_i^*)^2 \tag{6.18}$$

The maximum of the log-likelihood is now obtained by setting this derivative to zero. From that we get the solution to our ML estimation problem.

$$\sigma_{\text{hit}} = \sqrt{\frac{1}{|Z_{\text{hit}}|} \sum_{z_i \in Z_{\text{hit}}} (z_i - z_i^*)^2} \tag{6.19}$$

The estimation of the remaining intrinsic parameter λ_{short} proceeds just about in the same way. The posterior over the data Z_{short} is given by

$$p(Z_{\text{short}} \mid X, m, \Theta) = \prod_{z_i \in Z_{\text{short}}} p_{\text{short}}(z_i \mid x_i, m) \tag{6.20}$$

$$= \prod_{z_i \in Z_{\text{short}}} \lambda_{\text{short}} \, e^{-\lambda_{\text{short}} z_i}$$

The logarithm is

$$\begin{aligned} \log p(Z_{\text{short}} \mid X, m, \Theta) &= \sum_{z_i \in Z_{\text{short}}} \log \lambda_{\text{short}} - \lambda_{\text{short}} z_i \\ &= |Z_{\text{short}}| \log \lambda_{\text{short}} - \lambda_{\text{short}} \sum_{z_i \in Z_{\text{short}}} z_i \end{aligned} \tag{6.21}$$

The first derivative of this expression with respect to the intrinsic parameter λ_{short} is as follows:

$$\frac{\partial \log p(Z_{\text{short}} \mid X, m, \Theta)}{\partial \lambda_{\text{short}}} = \frac{|Z_{\text{short}}|}{\lambda_{\text{short}}} - \sum_{z_i \in Z_{\text{short}}} z_i \tag{6.22}$$

Setting this to zero gives us the ML estimate for the intrinsic parameter λ_{short}

$$\lambda_{\text{short}} = \frac{|Z_{\text{short}}|}{\sum_{z_i \in Z_{\text{short}}} z_i} \tag{6.23}$$

This derivation assumed knowledge of the parameters c_i. We now extend it to the case where the c_i's are unknown. As we shall see, the resulting ML estimation problem lacks a closed-form solution. However, we can devise a technique that iterates two steps, one that calculates an expectation for the c_i's and one that computes the intrinsic model parameters under these expectations. As noted, the resulting algorithm is an instance of the *expectation maximization* algorithm, usually abbreviated as EM.

EM ALGORITHM

To derive EM, it will be beneficial to define the likelihood of the data Z first:

$$\begin{aligned} &\log p(Z \mid X, m, \Theta) \\ &= \sum_{z_i \in Z} \log p(z_i \mid x_i, m, \Theta) \\ &= \sum_{z_i \in Z_{\text{hit}}} \log p_{\text{hit}}(z_i \mid x_i, m) + \sum_{z_i \in Z_{\text{short}}} \log p_{\text{short}}(z_i \mid x_i, m) \\ &\quad + \sum_{z_i \in Z_{\text{max}}} \log p_{\text{max}}(z_i \mid x_i, m) + \sum_{z_i \in Z_{\text{rand}}} \log p_{\text{rand}}(z_i \mid x_i, m) \end{aligned} \tag{6.24}$$

This expression can be rewritten using the variables c_i:

$$\log p(Z \mid X, m, \Theta) = \sum_{z_i \in Z} I(c_i = \text{hit}) \log p_{\text{hit}}(z_i \mid x_i, m) \tag{6.25}$$

$$
\begin{aligned}
&+I(c_i = \text{short})\ \log p_{\text{short}}(z_i \mid x_i, m) \\
&+I(c_i = \text{max})\ \log p_{\text{max}}(z_i \mid x_i, m) \\
&+I(c_i = \text{rand})\ \log p_{\text{rand}}(z_i \mid x_i, m)
\end{aligned}
$$

where I is the indicator function. Since the values for c_i are unknown, it is common to integrate them out. Put differently, EM maximizes the expectation $E[\log p(Z \mid X, m, \Theta)]$, where the expectation is taken over the unknown variables c_i:

$$
\begin{aligned}
(6.26)\quad & E[\log p(Z \mid X, m, \Theta)] \\
&= \sum_i p(c_i = \text{hit})\ \log p_{\text{hit}}(z_i \mid x_i, m) + p(c_i = \text{short})\ \log p_{\text{short}}(z_i \mid x_i, m) \\
&\quad + p(c_i = \text{max})\ \log p_{\text{max}}(z_i \mid x_i, m) + p(c_i = \text{rand})\ \log p_{\text{rand}}(z_i \mid x_i, m) \\
&=: \sum_i e_{i,\text{hit}}\ \log p_{\text{hit}}(z_i \mid x_i, m) + e_{i,\text{short}}\ \log p_{\text{short}}(z_i \mid x_i, m) \\
&\quad + e_{i,\text{max}}\ \log p_{\text{max}}(z_i \mid x_i, m) + e_{i,\text{rand}}\ \log p_{\text{rand}}(z_i \mid x_i, m)
\end{aligned}
$$

With the definition of the variable e as indicated. This expression is maximized in two steps. In a first step, we consider the intrinsic parameters σ_{hit} and λ_{short} given and calculate the expectation over the variables c_i.

$$
(6.27)\quad \begin{pmatrix} e_{i,\text{hit}} \\ e_{i,\text{short}} \\ e_{i,\text{max}} \\ e_{i,\text{rand}} \end{pmatrix} := \begin{pmatrix} p(c_i = \text{hit}) \\ p(c_i = \text{short}) \\ p(c_i = \text{max}) \\ p(c_i = \text{rand}) \end{pmatrix} = \eta \begin{pmatrix} p_{\text{hit}}(z_i \mid x_i, m) \\ p_{\text{short}}(z_i \mid x_i, m) \\ p_{\text{max}}(z_i \mid x_i, m) \\ p_{\text{rand}}(z_i \mid x_i, m) \end{pmatrix}
$$

The normalizer is given by

$$
\begin{aligned}
(6.28)\quad \eta &= [\, p_{\text{hit}}(z_i \mid x_i, m) + p_{\text{short}}(z_i \mid x_i, m) \\
&\quad + p_{\text{max}}(z_i \mid x_i, m) + p_{\text{rand}}(z_i \mid x_i, m)\,]^{-1}
\end{aligned}
$$

This step is called the "E-step," indicating that we calculate expectations over the latent variables c_i. The remaining step is now straightforward, since the expectations decouple the dependencies between the different components of the sensor model. First, we note that the ML mixture parameters are simply the normalized expectations

$$
(6.29)\quad \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\text{max}} \\ z_{\text{rand}} \end{pmatrix} = |Z|^{-1} \sum_i \begin{pmatrix} e_{i,\text{hit}} \\ e_{i,\text{short}} \\ e_{i,\text{max}} \\ e_{i,\text{rand}} \end{pmatrix}
$$

The ML parameters σ_{hit} and λ_{short} are then obtained analogously, by replacing the hard assignments in (6.19) and (6.23) by soft assignments weighted by the expectations.

$$
(6.30)\quad \sigma_{\text{hit}} = \sqrt{\frac{1}{\sum_{z_i \in Z} e_{i,\text{hit}}} \sum_{z_i \in Z} e_{i,\text{hit}} (z_i - z_i^*)^2}
$$

and

$$
(6.31)\quad \lambda_{\text{short}} = \frac{\sum_{z_i \in Z} e_{i,\text{short}}}{\sum_{z_i \in Z} e_{i,\text{short}}\ z_i}
$$

6.3.4 Practical Considerations

In practice, computing the densities of all sensor readings can be quite involved from a computational perspective. For example, laser range scanners often return hundreds of values per scan, at a rate of several scans per second. Since one has to perform a ray casting operation for each beam of the scan and every possible pose considered, the integration of the whole scan into the current belief cannot always be carried out in real-time. One typical approach to solve this problem is to incorporate only a small subset of all measurements (e.g., 8 equally spaced measurements per laser range scan instead of 360). This approach has an important additional benefit. Since adjacent beams of a range scan are often not independent, the state estimation process becomes less susceptible to correlated noise in adjacent measurements.

When dependencies between adjacent measurements are strong, the ML model may make the robot overconfident and yield suboptimal results. One simple remedy is to replace $p(z_t^k \mid x_t, m)$ by a "weaker" version $p(z_t^k \mid x_t, m)^\alpha$ for $\alpha < 1$. The intuition here is to reduce, by a factor of α, the information extracted from a sensor measurement (the log of this probability is given by $\alpha \log p(z_t^k \mid x_t, m)$). Another possibility—which we will only mention here—is to learn the intrinsic parameters in the context of the application: For example, in mobile localization it is possible to train the intrinsic parameters via gradient descent to yield good localization results over multiple time steps. Such a multi-time step methodology is significantly different from the single time step ML estimator described above. In practical implementations it can yield superior results; see Thrun (1998a).

The main drain of computing time for beam-based models is the ray casting operation. The runtime costs of computing $p(z_t \mid x_t, m)$ can be substantially reduced by *pre-cashing* the ray casting algorithm, and storing the

result in memory—so that the ray casting operation can be replaced by a (much faster) table lookup. An obvious implementation of this idea is to decompose the state space into a fine-grained three-dimensional grid, and to pre-compute the ranges z_t^{k*} for each grid cell. This idea was already investigated in Chapter 4.1. Depending on the resolution of the grid, the memory requirements can be significant. In mobile robot localization, we find that pre-computing the range with a grid resolution of 15 centimeters and 2 degrees works well for indoor localization problems. It fits well into the RAM for moderate-sized computers, yielding speed-ups by an order of magnitude over the plain implementation that casts rays online.

6.3.5 Limitations of the Beam Model

The beam-based sensor model, while closely linked to the geometry and physics of range finders, suffers two major drawbacks.

In particular, the beam-based model exhibits a *lack of smoothness*. In cluttered environments with many small obstacles, the distribution $p(z_t^k \mid x_t, m)$ can be very unsmooth in x_t. Consider, for example, an environment with many chairs and tables (like a typical conference room). A robot like the ones shown in Chapter 1 will sense the legs of those obstacles. Obviously, small changes of a robot's pose x_t can have a tremendous impact on the correct range of a sensor beam. As a result, the measurement model $p(z_t^k \mid x_t, m)$ is highly discontinuous in x_t. The heading direction θ_t is particularly affected, since small changes in heading can cause large displacements in x-y-space at a range.

Lack of smoothness has two problematic consequences. First, any approximate belief representation runs the danger of missing the correct state, as nearby states might have drastically different posterior likelihoods. This poses constraints on the accuracy of the approximation which, if not met, increase the resulting error in the posterior. Second, hill climbing methods for finding the most likely state are prone to local minima, due to the large number of local maxima in such unsmooth models.

The beam-based model is also computational involved. Evaluating $p(z_t^k \mid x_t, m)$ for each single sensor measurement z_t^k involves ray casting, which is computationally expensive. As noted above, the problem can be partially remedied by pre-computing the ranges over a discrete grid in pose space. Such an approach shifts the computation into an initial off-line phase, with the benefit that the algorithm is faster at run time. However, the resulting tables are very large, since they cover a large three-dimensional space. Thus, pre-computing ranges is computationally expensive and requires substantial memory.

6.4 Likelihood Fields for Range Finders

6.4.1 Basic Algorithm

LIKELIHOOD FIELD

We will now describe an alternative model, called *likelihood field*, which overcomes these limitations. This model lacks a plausible physical explanation. In fact, it is an "ad hoc" algorithm that does not necessarily compute a conditional probability relative to any meaningful generative model of the physics of sensors. However, the approach works well in practice. The resulting posteriors are much smoother even in cluttered space, and the computation is more efficient.

The key idea is to first project the end points of a sensor scan z_t into the global coordinate space of the map. To do so, we need to know where relative to the global coordinate frame the robot's local coordinate system is located, where on the robot the sensor beam z_k originates, and where the sensor points. As usual let $x_t = (x \;\; y \;\; \theta)^T$ denote a robot pose at time t. Keeping with our two-dimensional view of the world, we denote the relative location of the sensor in the robot's fixed, local coordinate system by $(x_{k,\text{sens}} \;\; y_{k,\text{sens}})^T$, and the angular orientation of the sensor beam relative to the robot's heading direction by $\theta_{k,\text{sens}}$. These values are sensor-specific. The end point of the measurement z_t^k is now mapped into the global coordinate system via the obvious trigonometric transformation.

$$\begin{pmatrix} x_{z_t^k} \\ y_{z_t^k} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{k,\text{sens}} \\ y_{k,\text{sens}} \end{pmatrix} + z_t^k \begin{pmatrix} \cos(\theta + \theta_{k,\text{sens}}) \\ \sin(\theta + \theta_{k,\text{sens}}) \end{pmatrix} \tag{6.32}$$

These coordinates are only meaningful when the sensor detects an obstacle. If the range sensor takes on its maximum value $z_t^k = z_{\max}$, these coordinates have no meaning in the physical world (even though the measurement does carry information). The likelihood field measurement model simply discards max-range readings.

Similar to the beam model discussed before, we assume three types of sources of noise and uncertainty:

1. **Measurement noise.** Noise arising from the measurement process is modeled using Gaussians. In x-y-space, this involves finding the nearest obstacle in the map. Let $dist$ denote the Euclidean distance between the

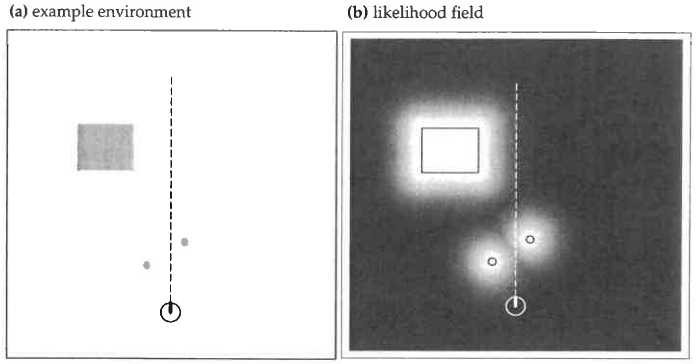


Figure 6.8 (a) Example environment with three obstacles (gray). The robot is located towards the bottom of the figure, and takes a measurement z_t^k as indicated by the dashed line. (b) Likelihood field for this obstacle configuration: the darker a location, the less likely it is to perceive an obstacle there. The probability $p(z_t^k \mid x_t, m)$ for the specific sensor beam is shown in Figure 6.9.

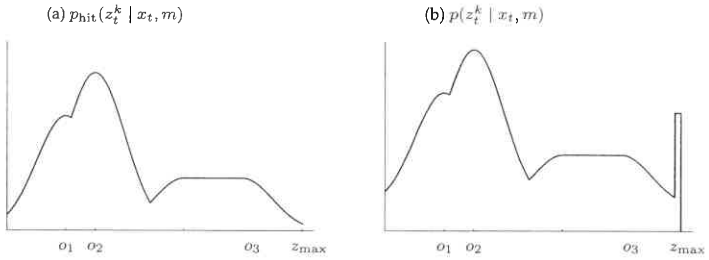


Figure 6.9 (a) Probability $p_{\text{hit}}(z_t^k)$ as a function of the measurement z_t^k, for the situation depicted in Figure 6.8. Here the sensor beam passes by three obstacles, with respective nearest points o_1, o_2, and o_3. (b) Sensor probability $p(z_t^k \mid x_t, m)$, obtained for the situation depicted in Figure 6.8, obtained by adding two uniform distributions.

measurement coordinates $(x_{z_t^k} \;\; y_{z_t^k})^T$ and the nearest object in the map m. Then the probability of a sensor measurement is given by a zero-centered Gaussian, which captures the sensor noise:

$$p_{\text{hit}}(z_t^k \mid x_t, m) \;=\; \varepsilon_{\sigma_{\text{hit}}}(dist) \tag{6.33}$$

Figure 6.8a depicts a map, and Figure 6.8b shows the corresponding Gaussian likelihood for measurement points $(x_{z_t^k} \;\; y_{z_t^k})^T$ in 2-D space. The brighter a location, the more likely it is to measure an object with a range finder. The density p_{hit} is now obtained by intersecting (and normalizing) the likelihood field by the sensor axis, indicated by the dashed line in Figure 6.8. The resulting function is the one shown in Figure 6.9a.

2. **Failures.** As before, we assume that max-range readings have a distinct large likelihood. As before, this is modeled by a point-mass distribution $p_{\max}$.

3. **Unexplained random measurements.** Finally, a uniform distribution p_{rand} is used to model random noise in perception.

Just as for the beam-based sensor model, the desired probability $p(z_t^k \mid x_t, m)$ integrates all three distributions:

$$z_{\text{hit}} \cdot p_{\text{hit}} + z_{\text{rand}} \cdot p_{\text{rand}} + z_{\max} \cdot p_{\max} \tag{6.34}$$

using the familiar mixing weights z_{hit}, z_{rand}, and $z_{\max}$. Figure 6.9b shows an example of the resulting distribution $p(z_t^k \mid x_t, m)$ along a measurement beam. It should be easy to see that this distribution combines p_{hit}, as shown in Figure 6.9a, and the distributions $p_{\max}$ and p_{rand}. Much of what we said about adjusting the mixing parameters transfers over to our new sensor model. They can be adjusted by hand or learned using the ML estimator. A representation like the one in Figure 6.8b, which depicts the likelihood of an obstacle detection as a function of global x-y-coordinates, is called the *likelihood field*.

Table 6.3 provides an algorithm for calculating the measurement probability using the likelihood field. The reader should already be familiar with the outer loop, which multiplies the individual values of $p(z_t^k \mid x_t, m)$, assuming independence between the noise in different sensor beams. Line 4 checks if the sensor reading is a max range reading, in which case it is simply ignored. Lines 5 to 8 handle the interesting case: Here the distance to the nearest obstacle in x-y-space is computed (line 7), and the resulting likelihood is obtained in line 8 by mixing a normal and a uniform distribution. As

1: **Algorithm likelihood_field_range_finder_model**(z_t, x_t, m)**:**

2: $q = 1$

3: *for all* k *do*

4: *if* $z_t^k \neq z_{\max}$

5: $x_{z_t^k} = x + x_{k,\text{sens}} \cos\theta - y_{k,\text{sens}} \sin\theta + z_t^k \cos(\theta + \theta_{k,\text{sens}})$

6: $y_{z_t^k} = y + y_{k,\text{sens}} \cos\theta + x_{k,\text{sens}} \sin\theta + z_t^k \sin(\theta + \theta_{k,\text{sens}})$

7: $dist = \min_{x',y'} \left\{ \sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} \,\middle|\, \langle x', y' \rangle \text{ occupied in } m \right\}$

8: $q = q \cdot \left(z_{\text{hit}} \cdot \textbf{prob}(dist, \sigma_{\text{hit}}) + \frac{z_{\text{random}}}{z_{\max}} \right)$

9: *return* q

Table 6.3 Algorithm for computing the likelihood of a range finder scan using Euclidean distance to the nearest neighbor. The function $\textbf{prob}(dist, \sigma_{\text{hit}})$ computes the probability of the distance under a zero-centered Gaussian distribution with standard deviation σ_{hit}.

before, the function $\textbf{prob}(dist, \sigma_{\text{hit}})$ computes the probability of $dist$ under a zero-centered Gaussian distribution with standard deviation σ_{hit}.

The search for the nearest neighbor in the map (line 7) is the most costly operation in algorithm **likelihood_field_range_finder_model**. To speed up this search, it is advantageous to pre-compute the likelihood field, so that calculating the probability of a measurement amounts to a coordinate transformation followed by a table lookup. Of course, if a discrete grid is used, the result of the lookup is only approximate, in that it might return the wrong obstacle coordinates. However, the effect on the probability $p(z_t^k \mid x_t, m)$ is typically small even for moderately course grids.

6.4.2 Extensions

A key advantage of the likelihood field model over the beam-based model discussed before is smoothness. Due to the smoothness of the Euclidean distance, small changes in the robot's pose x_t only have small effects on the resulting distribution $p(z_t^k \mid x_t, m)$. Another key advantage is that the pre-computation takes place in 2-D, instead of 3-D, increasing the compactness of the pre-computed information.

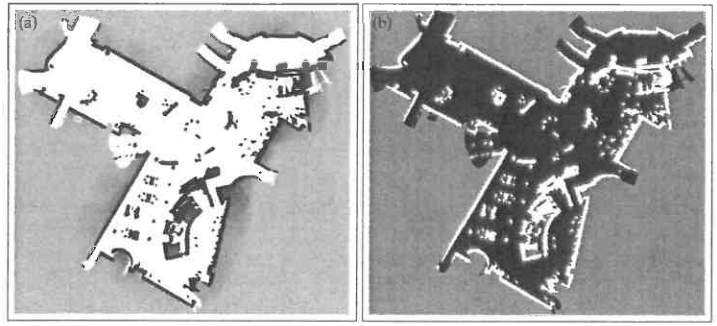


Figure 6.10 (a) Occupancy grid map of the San Jose Tech Museum, (b) pre-processed likelihood field.

However, the current model has three key disadvantages: First, it does not explicitly model people and other dynamics that might cause short readings. Second, it treats sensors as if they can "see through walls." This is because the ray casting operation was replaced by a nearest neighbor function, which is incapable of determining whether a path to a point is intercepted by an obstacle in the map. And third, our approach does not take map uncertainty into account. In particular, it cannot handle *unexplored* areas, for which the map is highly uncertain or unspecified.

The basic algorithm **likelihood_field_range_finder_model** can be extended to diminish the effect of these limitations. For example, one might sort map occupancy values into three categories: *occupied*, *free*, and *unknown*, instead of just the first two. When a sensor measurement z_t^k falls into the category *unknown*, its probability $p(z_t^k \mid x_t, m)$ is assumed to be the constant value $\frac{1}{z_{\max}}$. The resulting probabilistic model is crude. It assumes that in the unexplored space every sensor measurement is equally likely.

Figure 6.10 shows a map and the corresponding likelihood field. Here again the gray-level of an x-y-location indicates the likelihood of receiving a sensor reading there. The reader may notice that the distance to the nearest obstacle is only employed *inside* the map, which corresponds to the explored terrain. Outside, the likelihood $p(z_t^k \mid x_t, m)$ is a constant. For computational efficiency, it is worthwhile to pre-compute the nearest neighbor for a fine-grained 2-D grid.

Likelihood fields over the visible space can also be defined for the most

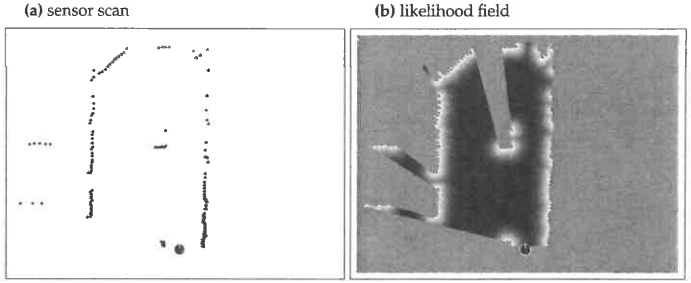


Figure 6.11 (a) Sensor scan, from a bird's eye perspective. The robot is placed at the bottom of this figure, generating a proximity scan that consists of the 180 dots in front of the robot. (b) Likelihood function generated from this sensor scan. The darker a region, the smaller the likelihood for sensing an object there. Notice that occluded regions are white, hence infer no penalty.

recent scan, which in fact defines a local map. Figure 6.11 shows such a likelihood field. It plays an important role in techniques that align individual scans.

6.5 Correlation-Based Measurement Models

There exist a number of range sensor models in the literature that measure correlations between a measurement and the map. A common technique is known as *map matching*. Map matching requires techniques discussed in later chapters of this book, namely the ability to transform scans into occupancy maps. Typically, map matching compiles small numbers of consecutive scans into *local maps*, denoted m_{local}. Figure 6.12 shows such a local map, here in the form of an occupancy grid map. The sensor measurement model compares the local map m_{local} to the global map m, such that the more similar m and m_{local}, the larger $p(m_{\text{local}} \mid x_t, m)$. Since the local map is represented relative to the robot location, this comparison requires that the cells of the local map are transformed into the coordinate framework of the global map. Such a transformation can be done similar to the coordinate transform (6.32) of sensor measurements used in the likelihood field model. If the robot is at location x_t, we denote by $m_{x,y,\text{local}}(x_t)$ the grid cell in the local map that

MAP MATCHING

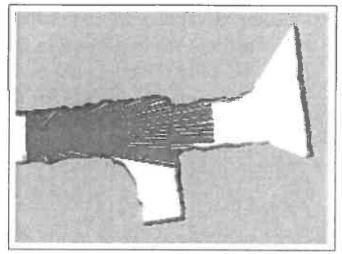


Figure 6.12 Example of a local map generated from 10 range scans, one of which is shown.

corresponds to $(x \;\; y)^T$ in global coordinates. Once both maps are in the same reference frame, they can be compared using the map correlation function, which is defined as follows:

$$\rho_{m,m_{\text{local}},x_t} = \frac{\sum_{x,y}(m_{x,y} - \bar{m}) \cdot (m_{x,y,\text{local}}(x_t) - \bar{m})}{\sqrt{\sum_{x,y}(m_{x,y} - \bar{m})^2 \sum_{x,y}(m_{x,y,\text{local}}(x_t) - \bar{m})^2}} \tag{6.35}$$

Here the sum is evaluated over cells defined in both maps, and $\bar{m}$ is the average map value:

$$\bar{m} = \frac{1}{2N}\sum_{x,y}(m_{x,y} + m_{x,y,\text{local}}) \tag{6.36}$$

where N denotes the number of elements in the overlap between the local and global map. The correlation $\rho_{m,m_{\text{local}},x_t}$ scales between ± 1. Map matching interprets the value

$$p(m_{\text{local}} \mid x_t, m) = \max\{\rho_{m,m_{\text{local}},x_t}, 0\} \tag{6.37}$$

as the probability of the local map conditioned on the global map m and the robot pose x_t. If the local map is generated from a single range scan z_t, this probability substitutes the measurement probability $p(z_t \mid x_t, m)$.

Map matching has a number of nice properties: just like the likelihood field model, it is easy to compute, though it does not yield smooth probabilities in the pose parameter x_t. One way to approximate the likelihood field (and to obtain smoothness) is to convolve the map m with a Gaussian smoothness kernel, and to run map matching on this smoothed map.

A key advantage of map matching over likelihood fields is that it explicitly considers the free-space in the scoring of two maps; the likelihood field technique only considers the end point of the scans, which by definition correspond to occupied space (or noise). On the other hand, many mapping techniques build local maps beyond the reach of the sensors. For example, many techniques build circular maps around the robot, setting to 0.5 areas beyond the range of actual sensor measurements. In such cases, there is a danger that the result of map matching incorporates areas beyond the actual measurement range, as if the sensor can "see through walls." Such side-effects are found in a number of implemented map matching techniques.

A further disadvantage is that map matching does not possess a plausible physical explanation. Correlations are the normalized quadratic distance between maps, which is *not* the noise characteristic of range sensors.

6.6 Feature-Based Measurement Models

6.6.1 Feature Extraction

FEATURES

The sensor models discussed thus far are all based on raw sensor measurements. An alternative approach is to extract *features* from the measurements. If we denote the feature extractor as a function f, the features extracted from a range measurement are given by $f(z_t)$. Most feature extractors extract a small number of features from high-dimensional sensor measurements. A key advantage of this approach is the enormous reduction of computational complexity: While inference in the high-dimensional measurement space can be costly, inference in the low-dimensional feature space can be orders of magnitude more efficient.

The discussion of specific algorithms for feature extraction is beyond the scope of this book. The literature offers a wide range of features for a number of different sensors. For range sensors, it is common to identify lines, corners, or local minima in range scans, which correspond to walls, corners, or objects such as tree trunks. When cameras are used for navigation, the processing of camera images falls into the realm of computer vision. Computer vision has devised a myriad of feature extraction techniques from camera images. Popular features include edges, corners, distinct patterns, and objects of distinct appearance. In robotics, it is also common to define places as features, such as hallways and intersections.

6.6.2 Landmark Measurements

In many robotics applications, features correspond to distinct objects in the physical world. For example, in indoor environments features may be door posts or windowsills; outdoors they may correspond to tree trunks or corners of buildings. In robotics, it is common to call those physical objects *landmarks*, to indicate that they are being used for robot navigation.

LANDMARKS

The most common model for processing landmarks assumes that the sensor can measure the range and the bearing of the landmark relative to the robot's local coordinate frame. Such sensors are called *range and bearing sensors*. The existence of a range-bearing sensor is not an implausible assumption: Any local feature extracted from range scans come with range and bearing information, as do visual features detected by stereo vision. In addition, the feature extractor may generate a *signature*. In this book, we assume a signature is a numerical value (e.g., an average color); it may equally be an integer that characterizes the type of the observed landmark, or a multi-dimensional vector characterizing a landmark (e.g., height and color).

RANGE AND BEARING SENSOR

SIGNATURE OF A LANDMARK

If we denote the range by r, the bearing by ϕ, and the signature by s, the feature vector is given by a collection of triplets

$$f(z_t) = \{f_t^1, f_t^2, \ldots\} = \{\begin{pmatrix} r_t^1 \\ \phi_t^1 \\ s_t^1 \end{pmatrix}, \begin{pmatrix} r_t^2 \\ \phi_t^1 \\ s_t^2 \end{pmatrix}, \ldots\} \tag{6.38}$$

The number of features identified at each time step is variable. However, many probabilistic robotic algorithms assume conditional independence between features

$$p(f(z_t) \mid x_t, m) = \prod_i p(r_t^i, \phi_t^i, s_t^i \mid x_t, m) \tag{6.39}$$

Conditional independence applies if the noise in each individual measurement $(r_t^i \quad \phi_t^i \quad s_t^i)^T$ is independent of the noise in other measurements $(r_t^j \quad \phi_t^j \quad s_t^j)^T$ (for $i \neq j$). Under the conditional independence assumption, we can process one feature at a time, just as we did in several of our range measurement models. This makes it much easier to develop algorithms that implement probabilistic measurement models.

Let us now devise a sensor model for features. In the beginning of this chapter, we distinguished between two types of maps: *feature-based* and *location-based*. Landmark measurement models are usually defined only for feature-based maps. The reader may recall that those maps consist of lists of features, $m = \{m_1, m_2, \ldots\}$. Each feature may possess a signature and a

location coordinate. The location of a feature, denoted $m_{i,x}$ and $m_{i,y}$, is simply its coordinate in the global coordinate frame of the map.

The measurement vector for a noise-free landmark sensor is easily specified by the standard geometric laws. We will model noise in landmark perception by independent Gaussian noise on the range, bearing, and the signature. The resulting measurement model is formulated for the case where the i-th feature at time t corresponds to the j-th landmark in the map. As usual, the robot pose is given by $x_t = (x \;\; y \;\; \theta)^T$.

$$\begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \\ s_j \end{pmatrix} + \begin{pmatrix} \varepsilon_{\sigma_r^2} \\ \varepsilon_{\sigma_\phi^2} \\ \varepsilon_{\sigma_s^2} \end{pmatrix} \tag{6.40}$$

Here ε_{σ_r}, $\varepsilon_{\sigma_\phi}$, and ε_{σ_s} are zero-mean Gaussian error variables with standard deviations σ_r, σ_ϕ, and σ_s, respectively.

6.6.3 Sensor Model with Known Correspondence

DATA ASSOCIATION PROBLEM

A key problem for range/bearing sensors is known as the *data association problem*. This problem arises when landmarks cannot be uniquely identified, so that some residual uncertainty exists with regards to the identity of a landmark. For developing a range/bearing sensor model, it shall prove useful to introduce a *correspondence variable* between the feature f_t^i and the landmark m_j in the map. This variable will be denoted by c_t^i with $c_t^i \in \{1, \ldots, N+1\}$; N is the number of landmarks in the map m. If $c_t^i = j \leq N$, then the i-th feature observed at time t corresponds to the j-th landmark in the map. In other words, c_t^i is the true identity of an observed feature. The only exception occurs with $c_t^i = N + 1$: Here a feature observation does not correspond to any feature in the map m. This case is important for handling spurious landmarks; it is also of great relevance for the topic of robotic mapping, in which the robot may encounter previously unobserved landmarks.

CORRESPONDENCE VARIABLE

Table 6.4 depicts the algorithm for calculating the probability of a feature f_t^i with known correspondence $c_t^i \leq N$. Lines 3 and 4 calculate the true range and bearing to the landmark. The probability of the measured ranges and bearing is then calculated in line 5, assuming independence in the noise. As the reader easily verifies, this algorithm implements Equation (6.40).

1:	**Algorithm landmark_model_known_correspondence**(f_t^i, c_t^i, x_t, m)**:**
2:	$j = c_t^i$
3:	$\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$
4:	$\hat{\phi} = \text{atan2}(m_{j,y} - y, m_{j,x} - x)$
5:	$q = \textbf{prob}(r_t^i - \hat{r}, \sigma_r) \cdot \textbf{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi) \cdot \textbf{prob}(s_t^i - s_j, \sigma_s)$
6:	*return* q

Table 6.4 Algorithm for computing the likelihood of a landmark measurement. The algorithm requires as input an observed feature $f_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$, and the true identity of the feature c_t^i, the robot pose $x_t = (x \;\; y \;\; \theta)^T$, and the map m. Its output is the numerical probability $p(f_t^i \mid c_t^i, m, x_t)$.

6.6.4 Sampling Poses

Sometimes it is desirable to sample robot poses x_t that correspond to a measurement f_t^i with feature identity c_t^i. We already encountered such sampling algorithms in the previous chapter, where we discussed robot motion models. Such sampling models are also desirable for sensor models. For example, when localizing a robot globally, it shall become useful to generate sample poses that incorporate a sensor measurement to generate initial guesses for the robot pose.

While in the general case, sampling poses x_t that correspond to a sensor measurement z_t is difficult, for our landmark model we can actually provide an efficient sampling algorithm. However, such sampling is only possible under further assumptions. In particular, we have to know the prior $p(x_t \mid c_t^i, m)$. For simplicity, let us assume this prior is uniform (it generally is not!). Bayes rule then suggests that

$$\begin{aligned} p(x_t \mid f_t^i, c_t^i, m) &= \eta \, p(f_t^i \mid c_t^i, x_t, m) \, p(x_t \mid c_t^i, m) \\ &= \eta \, p(f_t^i \mid c_t^i, x_t, m) \end{aligned} \tag{6.41}$$

Sampling from $p(x_t \mid f_t^i, c_t^i, m)$ can now be achieved from the "inverse" of the sensor model $p(f_t^i \mid c_t^i, x_t, m)$. Table 6.5 depicts an algorithm that samples poses x_t. The algorithm is tricky: Even in the noise-free case, a landmark observation does not uniquely determine the location of the robot. Instead, the robot may be on a circle around the landmark, whose diameter is the range to the landmark. The indeterminacy of the robot pose also follows

1: **Algorithm sample_landmark_model_known_correspondence**(f_t^i, c_t^i, m)**:**

2: $j = c_t^i$

3: $\hat{\gamma} = \text{rand}(0, 2\pi)$

4: $\hat{r} = r_t^i + \textbf{sample}(\sigma_r)$

5: $\hat{\phi} = \phi_t^i + \textbf{sample}(\sigma_\phi)$

6: $x = m_{j,x} + \hat{r} \cos \hat{\gamma}$

7: $y = m_{j,y} + \hat{r} \sin \hat{\gamma}$

8: $\theta = \hat{\gamma} - \pi - \hat{\phi}$

9: *return* $(x \ y \ \theta)^T$

Table 6.5 Algorithm for sampling poses from a landmark measurement $f_t^i = (r_t^i \ \phi_t^i \ s_t^i)^T$ with known identity c_t^i.

from the fact that the range and bearing provide two constraints in a three-dimensional space of robot poses.

To implement a pose sampler, we have to sample the remaining free parameter, which determines where on the circle around the landmark the robot is located. This parameter is called $\hat{\gamma}$ in Table 6.5, and is chosen at random in line 3. Lines 4 and 5 perturb the measured range and bearing, exploiting the fact that the mean and the measurement are treated symmetrically in Gaussians. Finally, lines 6 through 8 recover the pose that corresponds to $\hat{\gamma}$, $\hat{r}$, and $\hat{\phi}$.

Figure 6.13 illustrates the pose distribution $p(x_t \mid f_t^i, c_t^i, m)$ (left diagram) and also shows a sample drawn with our algorithm **sample_landmark_model_known_correspondence** (right diagram). The posterior is projected into x-y-space, where it becomes a ring around the measured range r_t^i. In 3-D pose space, it is a spiral that unfolds the ring with the angle θ.

6.6.5 Further Considerations

Both algorithms for landmark-based measurements assume known correspondence. The case of unknown correspondence will be discussed in detail in later chapters, when we address algorithms for localization and mapping under unknown correspondence.

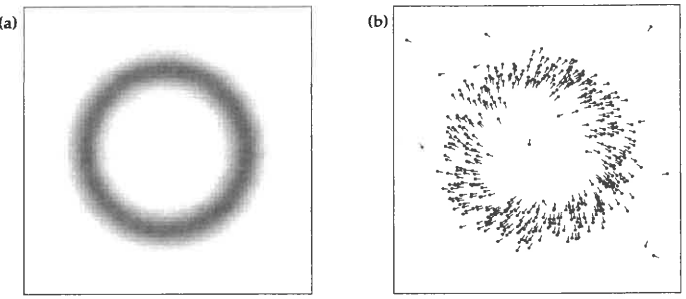


Figure 6.13 Landmark detection model: (a) Posterior distribution of the robot's pose given that it detected a landmark in 5m distance and 30deg relative bearing (projected onto 2-D). (b) Sample robot poses generated from such a detection. The lines indicate the orientation of the poses.

A comment is in order on the topic of landmark signature. Most published algorithms do not make the use of appearance features explicit. When the signature is not provided, all landmarks look equal, and the data association problem of estimating the correspondence variables is harder. We have included the signature in our model because it is a valuable source of information that can often be easily extracted from the sensor measurements.

As noted above, the main motivation for using features instead of the full measurement vector is computational in nature: It is much easier to manage a few hundred features than a few billion range measurements. Our model presented here is extremely crude, and it clearly does not capture the physical laws that underlie the sensor formation process. Nevertheless, the model tends to work well in a great number of applications.

It is important to notice that the reduction of measurements into features comes at a price. In the robotics literature, features are often (mis)taken for *sufficient statistics* of the measurement vector z_t. Let X be a variable of interest (e.g., a map, a pose), and Y some other information that we might bring to bear (e.g., past sensor measurements). Then f is a sufficient statistics of z_t if

SUFFICIENT STATISTICS

$$p(X \mid z_t, Y) \quad = \quad p(X \mid f(z_t), Y) \tag{6.42}$$

In practice, however, a lot of information is sacrificed by using features instead of the full measurement vector. This lost information makes certain

problems more difficult, such as the data association problem of determining whether or not the robot just revisited a previously explored location. It is easy to understand the effects of feature extraction by introspection: When you open your eyes, the visual image of your environment is probably sufficient to tell you unambiguously where you are—even if you were globally uncertain before. If, on the other hand, you only sense certain features, such as the relative location of door posts and windowsills, you would probably be much less certain as to where you are. Quite likely the information may be insufficient for global localization.

With the advent of fast computers, features have gradually lost importance in the field of robotics. Especially when using range sensors, most state-of-the-art algorithms rely on dense measurement vectors, and they use dense location-based maps to represent the environment. Nevertheless, features are still great for educational purposes. They enable us to introduce the basic concepts in probabilistic robotics, and with proper treatment of problems such as the correspondence problem they can be brought to bear even in cases where maps are composed of dense sets of scan points. For this reason, a number of algorithms in this book are first described for feature representations, and then extended into algorithms using raw sensor measurements.

6.7 Practical Considerations

This section surveyed a range of measurement models. We placed a strong emphasis on models for range finders, due to their great importance in robotics. However, the models discussed here are only representatives of a much broader class of probabilistic models. In choosing the right model, it is important to trade off physical realism with properties that might be desirable for an algorithm using these models. For example, we noted that a physically realistic model of range sensors may yield probabilities that are not smooth in the alleged robot pose—which in turn causes problems for algorithms such as particle filters. Physical realism is therefore not the only criterion in choosing the right sensor model; an equally important criterion is the goodness of a model for the algorithm that utilizes it.

As a general rule of thumb, the more accurate a model, the better. In particular, the more information we can extract from a sensor measurement, the better. Feature-based models extract relatively little information, by virtue of the fact that feature extractors project high-dimensional sensor measurements into lower dimensional space. As a result, feature-based methods tend

to produce inferior results. This disadvantage is offset by superior computational properties of feature-based representations.

When adjusting the intrinsic parameters of a measurement model, it is often useful to artificially inflate the uncertainty. This is because of a key limitation of the probabilistic approach: to make probabilistic techniques computationally tractable, we have to ignore dependencies that exist in the physical world, along with a myriad of latent variables that cause these dependencies. When such dependencies are not modeled, algorithms that integrate evidence from multiple measurements quickly become overconfident. Such *overconfidence* can ultimately lead to wrong conclusions, which negatively affects the results. In practice, it is therefore a good rule of thumb to reduce the information conveyed by a sensor. Doing so by projecting the measurement into a low-dimensional feature space is one way of achieving this. However, it suffers the limitations mentioned above. Uniformly decaying the information by exponentiating a measurement model with a parameter α, as discussed in Chapter 6.3.4, is a much better way, in that it does not introduce additional variance in the outcome of a probabilistic algorithm.

OVERCONFIDENCE

6.8 Summary

This section described probabilistic measurement models.

- Starting with models for range finders—and lasers in particular—we discussed measurement models $p(z_t^k \mid x_t, m)$. The first such model used ray casting to determine the shape of $p(z_t^k \mid x_t, m)$ for particular maps m and poses x_t. We devised a mixture model that addressed the various types of noise that can affect range measurements.

- We devised a maximum likelihood technique for identifying the intrinsic noise parameters of the measurement model. Since the measurement model is a mixture model, we provided an iterative procedure for maximum likelihood estimation. Our approach was an instance of the expectation maximization algorithm, which alternates a phase that calculates expectations over the type of error underlying a measurement, with a maximization phase that finds in closed form the best set of intrinsic parameters relative to these expectations.

- An alternative measurement model for range finders is based on likelihood fields. This technique used the nearest distance in 2-D coordinates to model the probability $p(z_t^k \mid x_t, m)$. We noted that this approach tends

to yield smoother distributions $p(z_t^k \mid x_t, m)$. This comes at the expense of undesired side effects: The likelihood field technique ignores information pertaining to free-space, and it fails to consider occlusions in the interpretation of range measurements.

- A third measurement model is based on map matching. Map matching maps sensor scans into local maps, and correlates those maps with global maps. This approach lacks a physical motivation, but can be implemented very efficiently.
- We discussed how pre-computation can reduce the computational burden at runtime. In the beam-based measurement model, the pre-computation takes place in 3-D; the likelihood field requires only a 2-D pre-computation.
- We presented a feature-based sensor model, in which the robot extracts the range, bearing, and signature of nearby landmarks. Feature-based techniques extract from the raw sensor measurements distinct features. In doing so, they reduce the dimensionality of the sensor measurements by several orders of magnitude.
- At the end of the chapter, a discussion on practical issues pointed out some of the pitfalls that may arise in concrete implementations.

6.9 Bibliographical Remarks

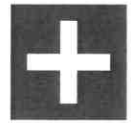
This chapter only skims the rich literature on physical modeling of sensors. More accurate models of sonar range sensor can be found in Blahut et al. (1991); Grunbaum et al. (1992) and in Etter (1996). Models of laser range finders are described by Rees (2001). An empirical discussion of proper noise models can be found in Sahin et al. (1998). Relative to these models, the models in this chapter are extremely crude.

An early work on beam models for range sensors can be found in the seminal work by Moravec (1988). A similar model was later applied to mobile robot localization by Burgard et al. (1996). A beam-based model like the one described in this chapter together with the pre-caching of range measurements has first been described by Fox et al. (1999b). The likelihood fields have first been published by Thrun (2001), although they are closely related to the rich literature on scan matching techniques (Besl and McKay 1992). They in fact can be regarded as a soft variant of the correlation model described by Konolige and Chou (1999). Methods for computing the correlation between occupancy grid maps have also been quite popular. Thrun (1993) computes the sum of squared errors between the individual cells of two grid maps. Schiele and Crowley (1994) present a comparison of different models including correlation-based approaches. Yamauchi and Langley (1997) analyzed the robustness of map matching to dynamic environments. Duckett and Nehmzow (2001) transform local occupancy grids into histograms that can be matched more efficiently.

Range and bearings measurements for point landmarks are commonplace in the SLAM literature. Possibly the first mention is by Leonard and Durrant-Whyte (1991). In earlier work, Crowley (1989) devised measurement models for straight line objects.

6.10 Exercises

1. Many early robots navigating using features used artificial landmarks in the environment that were easy to recognize. A good place to mount such markers is a ceiling (why?). A classical example is a visual marker: Suppose we attach the following marker to the ceiling:



Let the world coordinates of the marker be x_m and y_m, and its orientation relative to the global coordinate system θ_m. We will denote the robot's pose by x_r, y_r, and θ_r.

Now assume that we are given a routine that can detect the marker in the image plane of a perspective camera. Let x_i and y_i denote the coordinates of the marker in the image plane, and θ_i its angular orientation. The camera has a focal length of f. From projective geometry, we know that each displacement d in x-y-space gets projected to a proportional displacement of $d \cdot \frac{f}{h}$ in the image plane. (You have to make some choices on your coordinate systems; make these choices explicit).

Your questions:

(a) Describe mathematically where to expect the marker (in global coordinates x_m, y_m, θ_m) when its image coordinates are x_i, y_i, θ_i, and the robot is at x_r, y_r, θ_r.

(b) Provide a mathematical equation for computing the image coordinates x_i, y_i, θ_i from the robot pose x_r, y_r, θ_r and the marker coordinates x_m, y_m, θ_m.

(c) Now give a mathematical equation for determining the robot coordinates x_r, y_r, θ_r assuming we know the true marker coordinates x_m, y_m, θ_m and the image coordinates x_i, y_i, θ_i.