Multimedia Contents

5. Sensing and Estimation

Henrik I. Christensen, Gregory D. Hager

Sensing and estimation are essential aspects of the design of any robotic system. At a very basic level, the state of the robot itself must be estimated for feedback control. At a higher level, perception, which is defined here to be task-oriented interpretation of sensor data, allows the integration of sensor information across space and time to facilitate planning.

This chapter provides a brief overview of common sensing methods and estimation techniques that have found broad applicability in robotics. The presentation is structured according to a process model that includes sensing, feature extraction, data association, parameter estimation, and model integration. Several common sensing modalities are introduced and characterized. Common methods for estimation in linear and nonlinear systems are discussed, including statistical estimation, the Kalman filter, and sample-based methods. Strategies for robust estimation are also briefly described. Finally, several common representations for estimation are introduced.

5.1 Introduction

Controlling a robotic system would be relatively simple if a complete model of the environment was available, and if the robot actuators could execute motion commands perfectly relative to this model. Unfortunately, in most cases of interest, a complete world model is not available, and perfect control of mechanical structures is never a realistic assumption. Sensing and estimation are a means of compensating for this lack of complete information. Their role is to provide information about the state of the environment and the state of the robot system as a basis for control, decision making, and interaction with other agents in the environment, such as humans.

For the purposes of discussion, we will differentiate between sensing and estimation to recover the state of the robot itself, referred to as *proprioception*, versus sensing and estimation to recover the state of the external world, referred to as *exteroception*. In practice, most robot systems are designed to have the proprioception necessary to estimate and control their own physical state. On the other hand, recovering the state of the world from sensor data is usually a much larger and more complex problem.

Early work on computational perception for robotics assumed that one could recover a complete general-purpose model of the environment, use such a model to make decisions, and subsequently act on them, as for example presented by [5.1]. More recently it has become apparent that such an approach is not realistic. Indeed, considering that sensor-based robots now appear in diverse applications such as mobile surveillance, high-performance manipulation, and medical interventions, it is clear that appropriate sensing and estimation for a given system must be highly task dependent. Consequently, the discussion here is organized along the lines of task-oriented sensing and estimation of the external world.

Sensing and estimation together can be viewed as the process of transforming a physical quantity into a computer representation that can be used for further processing. Sensing is thus closely tied to transducers that transform some physical entity into a signal that can be processed by a computer. Sensing is also intimately tied to perception, the process of representing the sensory information in an task-oriented model of the world. However, sensor data is usually corrupted in various ways that complicate this process. Statistical noise arises from the transducer, discretization is introduced in the digitization process, and ambiguity is introduced by poor sensor selectivity to name a few examples. Estimation methods are thus introduced to support appropriate integration of information into models of the environment and for improvement of the signal-to-noise ratio.

In this chapter the general characteristics of sensing and estimation are introduced, while more in-depth presentations of select topics are provided in Part C of the handbook. In Sect. 5.2 the overall sensing/perception process is introduced. In Sect. 5.3 different kinds of sensors are introduced and some key characteristics are presented. Estimation of world representations can utilize a number of different methods involving both parametric and nonparametric techniques as discussed in Sect. 5.4. For model-based integration a variety of different representations can be used, as described in Sect. 5.5.

5.2 The Perception Process

The input to the perception process is typically twofold: (1) digital data from a number of sensors/transducers, and (2) a partial model of the environment (a world model) that includes information about the state of the robot and other relevant entities in the external world. The sensor data itself can take on a number of different forms such as a scalar or vector value $\boldsymbol{x}(\alpha, \beta)$ acquired over a time series $x(t)$, a scan $x_t(\theta_i)$, a vector field $\boldsymbol{x}$ or a three-dimensional volume $x(\rho, \theta, \phi)$. In many cases, a system must integrate data from several disparate sensors, for example, an estimate of the position of a mobile robot may integrate data from axis encoders, vision, global positioning system (GPS) data, and inertial sensors.

To further structure the discussion in this chapter, we adopt a general model of the perception process as shown in Fig. 5.1. In this model, we have included the most common operations applied to integrate sensor data with a world model. Depending on the task in question, some of the included modules may be missing, and others may themselves take on a complicated structure. However, the supplied model suffices to illustrate many of the issues in sensing and estimation. In the remainder of this section, we discuss an example from mobile localization to illustrate this model.

The initial problem in sensory processing is data preprocessing and feature extraction. The role of preprocessing is to reduce noise from the transducer, to remove any systematic errors, and to enhance relevant aspects of the data. In some cases, sensory information might also have to be temporally or spatially aligned for subsequent integration. There are innumerable ways

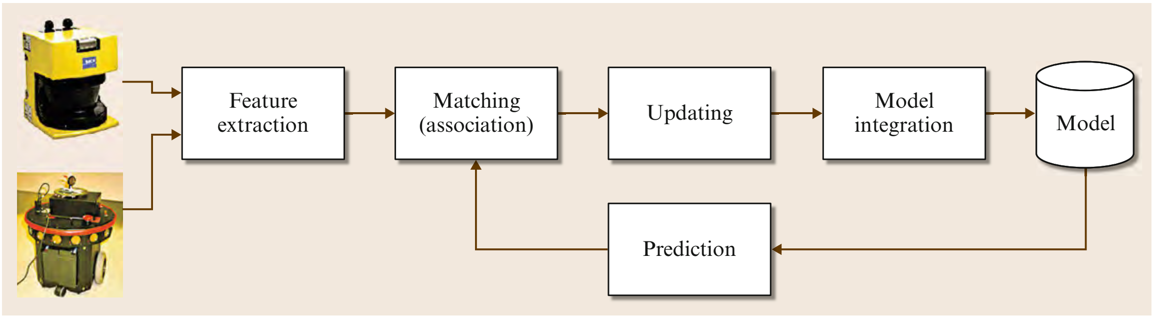


Fig. 5.1 Example of a perception process as discussed in this chapter

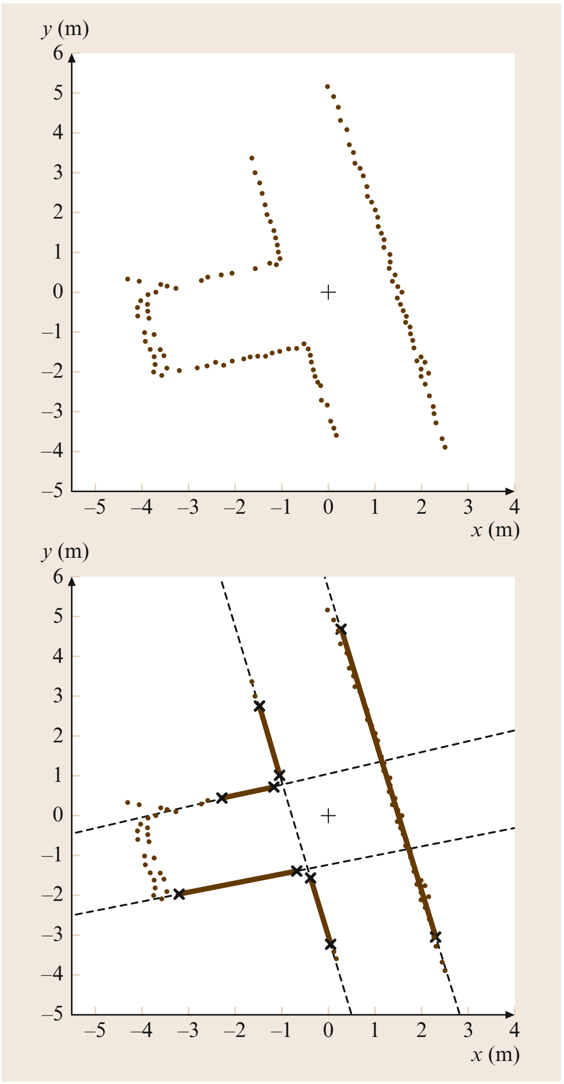


Fig. 5.2 An example of feature extraction from a laser scan (after [5.2])

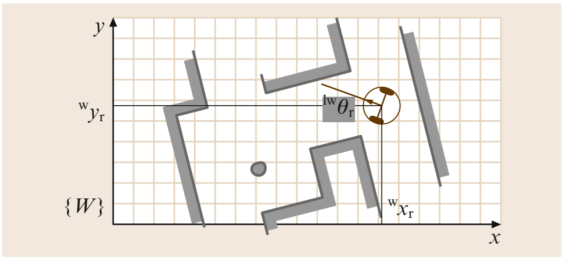

y
x
$^{\mathrm{w}}y_{\mathrm{r}}$
$^{\mathrm{w}}x_{\mathrm{r}}$
$^{\mathrm{w}}\theta_{\mathrm{r}}$
$\{W\}$

Fig. 5.3 An example environmental model for mobile robot localization (after [5.2])

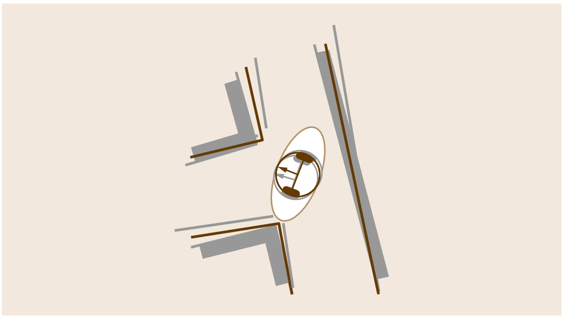


Fig. 5.4 Estimation of position and orientation for the example mobile robot (after [5.2])

that data can be preprocessed to enhance or extract features that are used in the integration. One common approach is model fitting, as illustrated for a laser scanner in Fig. 5.2. Once sensor information is available, it is often necessary to match the data with an existing model (Fig. 5.3). This model may be based on a priori known structure (e.g., a computer-aided design (CAD) model of the environment), or may have been built up from previously acquired data. Data association methods are commonly employed to estimate the relationship between sensor data and the model of the environment. In our mobile robot localization example, the extracted line features are matched against a polygonal world model. This matching process can be performed in several different ways, but in general it is an optimization that maximizes the alignment of features to the model.

Once sensory data has been matched against the world model it is possible to update the model with new information contained in the sensor data. In the example, the orientation and position of the robot relative to the world model can be updated (Fig. 5.4) from the matched line segments.

Finally, it may be possible to develop a dynamical system model of the underlying state being estimated. Using such a system model, it is possible to predict how the world changes over time until new sensory data is acquired. This can be used within a feed-forward prediction process, which in turn simplifies data association for new sensory readings, as shown in Fig. 5.1.

With this as a prologue, we now turn to discuss each step of the perception process in greater detail.

5.3 Sensors

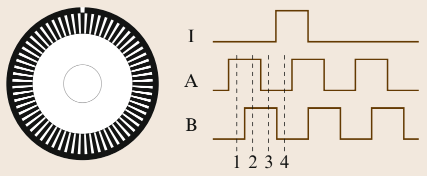
There are a variety of ways to classify sensors depending on what they measure, and how they measure it. As noted previously, proprioceptive sensors are used to measure the internal state of a robot, which might include position of different degrees of freedom, temperature, voltage on key components, motor current, force applied to an effector, and so forth. Exteroceptive sensors, on the other hand, generate information about the external environment in terms of distance to an object, interaction forces, tissue density, and so forth.

Sensors may also be differentiated based on whether they are passive or active. In general, an active sensor is one that emits energy into the environment, and measures properties of the environment based on the response. A passive sensor is one that is not active. Active sensors are generally more robust than passive sensors since they exert some control over the measured signal. For example, a passive stereo camera system must rely on the appearance of viewed surfaces when performing feature matching for triangulation (Chap. 31), whereas structured light systems project a pattern onto the scene and are thus less sensitive to scene characteristics. Even so, absorbtion, scattering or interference of the emitted signal can affect the performance of active sensors.

Proprioceptive sensors are typically passive and usually measure physical properties of the robot such as joint position, velocity, or acceleration, motor torque, and so forth. Exteroceptive sensors, on the other hand, can be further divided into contact and noncontact sensing. The contact sensors are typically the same modalities as used for proprioception, while noncontact sensor sensors involve most of the modalities that can be used for estimation of physical properties at a distance including intensity, range, direction, size, and so forth.

A classification of typical sensors according to method and typical application is shown in Table 5.1. More detail on methods of sensing, characterization of sensors, and general applications can for example be found in the Handbook of Modern Sensors [5.3] and in Part C of this handbook.

Estimation of rotational motion is fundamental to control of robot manipulators and also for estimation of ego-motion for mobile systems. The most common sensor for measurement of rotation is the quadrature encoder. It is composed of a transparent disc, with two periodic patterns that are out of phase, as shown in Fig. 5.5. Through the use of counters it is possible



State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

Fig. 5.5 Sketch of the quadrature encoder disc, and output from photodetectors placed over each of the two pattern. The corresponding state changes are shown on the *right*

Table 5.1 Classification of sensors frequently used in robotics according to sensing objective (proprioception (PC)/exteroception (EC)) and method (active/passive)

Classification	Sensor type	Sens	A/P
Tactile sensors	Switches/bumpers	EC	P
	Optical barriers	EC	A
	Proximity	EC	P/A
Haptic sensors	Contact arrays	EC	P
	Force/torque	PC/EC	P
	Resistive	EC	P
Motor/axis sensors	Brush encoders	PC	P
	Potentiometers	PC	P
	Resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacity encoders	EC	A
Heading sensors	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P
Beacon based	GPS	EC	A
(postion wrt	Active optical	EC	A
an inertial	Radio frequency (RF) beacons	EC	A
frame)	Ultrasound beacon	EC	A
	Reflective beacons	EC	A
Ranging	Capacitive sensor	EC	P
	Magnetic sensors	EC	P/A
	Camera	EC	P/A
	Sonar	EC	A
	Laser range	EC	A
	Structured light	EC	A
Speed/motion	Doppler radar	EC	A
	Doppler sound	EC	A
	Camera	EC	P
	Accelerometer	EC	P
Identification	Camera	EC	P
	Radio frequency identification RFID	EC	A
	Laser ranging	EC	A
	Radar	EC	A
	Ultrasound	EC	A
	Sound	EC	P

to directly compute the motion and its direction (the phasing between sensors A and B in Fig. 5.5). In addition the disc is frequently fitted with a single dot on the outer rim for indexing (specification of a zero index). The density of the pattern determines the resolution of the measurements. When fitting the sensor to a motor before a reduction gear it is easy to achieve accuracies beyond 1/1000°.

For the estimation of force and torque at an end-effector it is possible to use piezoelectric elements. These elements generate a voltage that is proportional to the introduced deformation. Through careful placement it is possible to measure both force and torque. The sensors are used in robotic manipulation as part of assembly systems, deburring, etc. and also in medical applications for the estimation of stress and contact. Force/torque sensors are widely available in a range of sizes and dynamic ranges, including new flexible arrays that can be mounted on a variety of end-effectors (Fig. 5.6). These arrays are more commonly referred to as *tactile sensors* as they begin to simulate the human tactile sense. See Chap. 19 and [5.4, 5] for recent reviews of the state of the art in tactile sensing. Potential problems with force sensors are a dead band on initial contact, and noisy data from the basic sensing elements, which calls for signal processing to clean up the data.

Ego-motion estimation is an important part of almost all robotic systems. To this end it is possible to use inertial measurement units (IMU). An IMU typically includes both accelerometers and gyros. Accelerometers are sensitive to all types of acceleration, which implies that both translation motion and rotation (centripetal forces) are measured in combination. Joint IMU units allow the estimation of rotation and translation, and allow for double integration to estimation the velocity, orientation, and position of a system, as for example reported in [5.6]. One of the problems associated with the use of an IMU is the need for double integration. Small biases and noise can result in significant divergence in the final estimate, which calls for use of detailed models and careful calibration and identification of sensor characteristics. An example of data from a cross-bow DMU-6x unit for a car driving on an unpaved road is shown in Fig. 5.7.

Much early work on mobile robotics, underwater robots, and some medical robotics relies on ultrasonic ranging. The general class of sensors are often termed sound navigation and ranging (sonars). The general principle is that the system emits a sound pulse and awaits the return of echoes that have bounced off objects in the environment. Knowing the transmission speed in the medium and the time of flight it is possible to compute the distance. The method was widely used in early robotics due to the availability of low-cost sensors with adequate performance. In underwater robotics this is still a primary sensor. Sonar is discussed in detail in Chap. 30.

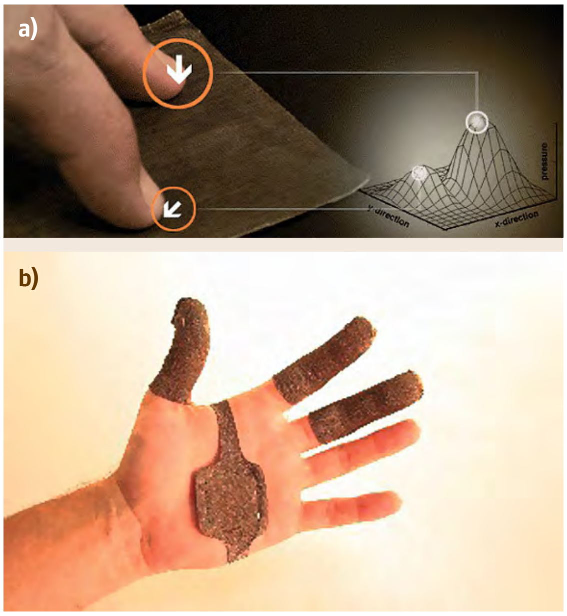


Fig. 5.6 (a) TactArray, a flexible capacitive array tactile sensor from Pressure Profile Systems, Inc., is appropriate for sensing contact locations and areas under sliding conditions. (b) Conformable TactArray sensors can fit on a human or robotic hand (courtesy Pressure Profile Systems, Inc.)

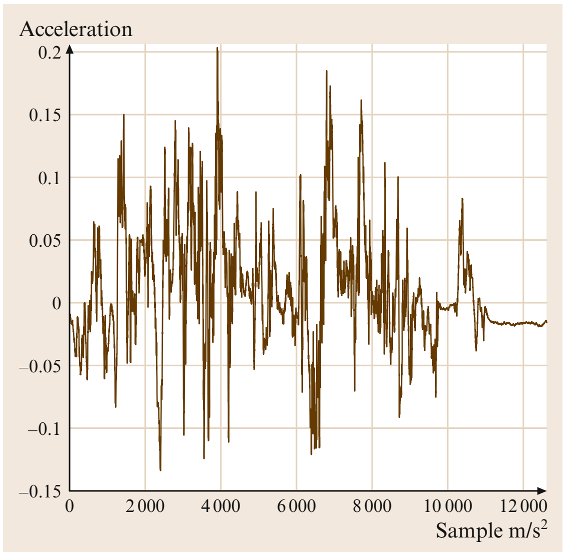


Fig. 5.7 Example data from an IMU unit for driving on an unpaved road

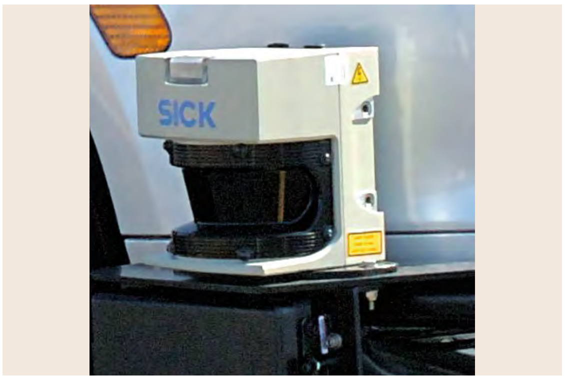


Fig. 5.8 Example laser ranging sensor (SICK LMS291) that is widely used in mobile robotics

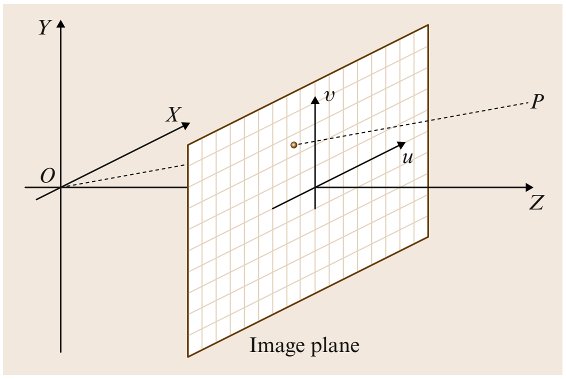


Fig. 5.9 The pinhole camera model

Recent progress on environmental modeling and navigation has, in many respects, been due to the emergence of low-cost high-fidelity laser scanning systems. The SICK series of laser scanners are time-of-flight scanners. The scanner sends out a pulse of light and measures the time to return. The standard scanner enables estimation of distances up to 80 m at centimeter or millimeter accuracy. The scanner measures distances in a plane with an angular resolution of 0.5–1°. The field of view is 180° resulting in 181–361 range measurements. The sensor data are contaminated by uniformly distributed noise, which must be considered in the detection of features or integration of data into a raw sensor map (Fig. 5.8).

Imaging sensors are a rich source of information for sensing and estimation. Imaging sensors come in a wide variety of configurations, varying according to imaging geometry, image resolution, sensor technology and the range of sensed spectral bands. Most readers are no doubt familiar with the traditional three-CCD, perspective color camera. In this case, there are three charge-coupled detector (CCD) arrays, each receiving a portion of the visible spectrum corresponding roughly to the human perception of red, green, and blue colors. A common and less expensive alternative is a so-called single-chip CCD camera. In this case, a special spatial array of color filters, usually referred to as a Bayer filter after its inventor Bryce Bayer, is employed. The resulting spatial array is subsequently processed (a process referred to as demosaicing) to provide color information for each pixel.

In the United States, image sensors traditionally contained 480 rows of 640 pixels according to the National Television System Committee (NTSC) standard created for analog transmission of television signals. The corresponding European standard, PAL, has 576 lines of 768 pixels. More recently, the advent of digital interfaces such as IEEE 1394 and USB 2.0 have allowed camera systems to be developed with significantly improved resolution ranging into the millions of pixels. At the same time, cost-effective infrared (IR) and ultraviolet (UV) cameras have become available, allowing the development of advanced multispectral image interpretation systems.

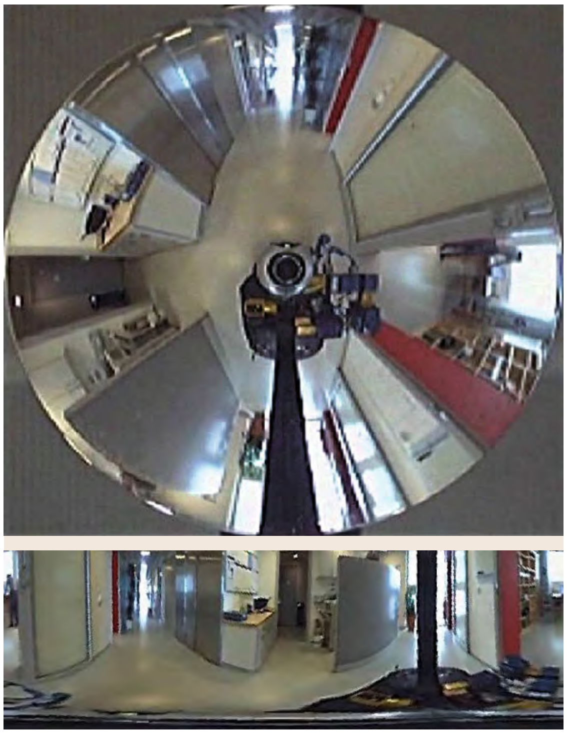


Fig. 5.10 Catadioptric image and the same image mapped to a cylindrical surface

A traditional imaging sensor contains an optical system that focuses light on a planar imaging array. In most cases, this system can be modeled using the classical pinhole camera model (shown in Fig. 5.9). Given a point $(x, y, z)^T$ in Euclidean space, the corresponding camera pixel coordinates $(u, v)^T$ are given by

$$(u - u_c) = \frac{f}{s_x}\frac{x}{z}\,,$$
$$(v - v_c) = \frac{f}{s_y}\frac{y}{z}\,, \qquad (5.1)$$

where f is the focal length of the lens system, u_c and v_c are the pixel coordinates of the center of projection, and s_x and s_y are the size of a single pixel on the imaging array. In practice, these models are also augmented with low-order models of image distortion. The values of these parameters for a given camera system can be determined experimentally using a variety of methods [5.7].

By combining a traditional perspective camera with a mirror, creating a so-called catadioptric system, it is possible to create imaging geometries that map fields of view as large as a hemisphere into a single image. Such systems are useful, for example, for surveillance, and their geometric properties provide for stable position referencing for mobile navigation [5.8]. An example image is shown in Fig. 5.10 together with the corresponding image when mapped to a cylindrical surface.

Active ranging cameras, which combine images from an optical camera with a dense range map, are now also widely available and quite cost effective. These systems perform triangulation between a light projector that throws a pattern with known structure into the scene, and a camera that views the scene and detects the pattern. By matching or correlating the pattern elements between the projector and the camera, and using the known relationship between them, it is possible to recover depth using standard methods. The camera/projector pair for depth recovery typically operate in the infrared to avoid the pattern being visible to the human eye. Thus, these systems typically include a third visible light camera to acquire a corresponding color image. Hence the name RGB-D (red–green–blue–depth) cameras. Fig. 5.11 shows two recent low-cost RGB-D products. In Fig. 5.12 is show the fused output for a tabletop scene.

The discussion above has touched on the most commonly employed robotic sensing devices. Many special-purpose sensors are employed for specific applications. In medicine (Chap. 63), ultrasound, X-ray, computed tomography, and magnetic resonance imaging are commonly employed. Underground mapping makes use of ground-penetrating radar [5.9]. Underwater robotics makes use of many variations on acoustical sensors. Further discussions of these more task-specific sensing modalities can be found in the application chapters in Part C of this handbook.

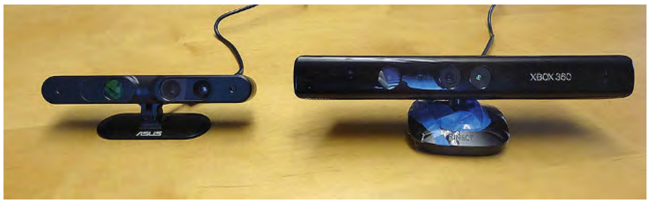


Fig. 5.11 Picture of proxim and kinect RGB-D sensors

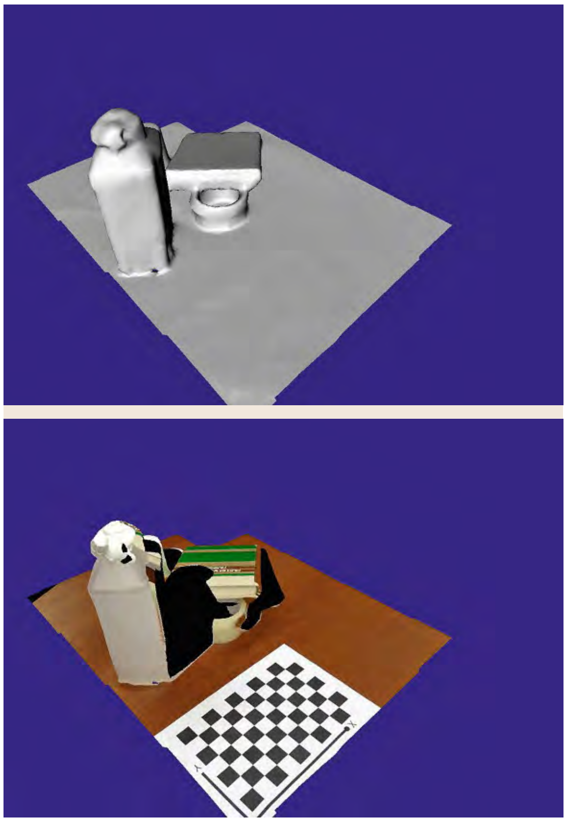


Fig. 5.12 Example of fused range images from RGB-D camera and the same scene as an intensity textured mesh

5.4 Estimation Processes

As discussed in the introduction, there are many different techniques for combining information from sensors. The appropriate set of techniques depends, to a great degree, on what is known a priori about the environment, what information is necessary for the task at hand, and what models for the sensing system are appropriate. Common methodologies include simple voting-based methods, parametric and nonparametric statistical estimation techniques, fuzzy logic-based systems, and Dempster–Shafer theory.

To illustrate this point, consider the robot localization problem introduced in Sect. 5.2. At the outset, if nothing is known about the environment, the robot may acquire a laser scan and try to produce an initial model of the environment using line segments. Since nothing is known a priori, the system must estimate:

1. The number of line segments
2. The data association between line segments and observed data values
3. The parameters of the line segments themselves.

This is a challenging problem that can be attacked by simple voting techniques such as the Hough transform [5.10] or random sample consensus (RANSAC) [5.11] or more sophisticated unsupervised clustering methods such as *k*-means [5.12], expectation maximization (EM) [5.13], or generalized principal component analysis (GPCA) [5.14]. In many cases, this is a computationally intensive, iterative process.

Conversely, if a prior CAD model for the environment is known, then the problem is to produce a small set of parameters (translation and rotation) of the model to match the data. This problem can be solved, using feature matching by aligning observed points to the model with iterative closest-point algorithms (ICP) [5.15] or other efficient combinatorial matching algorithms such as Monte Carlo methods [5.16]. The best method to apply again depends to a great degree on the structure of the environment and what is known a priori.

Once an initial registration is known, new data can take advantage of the fact that strong prior knowledge is available. In particular, as the robot moves, the sensor data should change in a predictable fashion. Thus is it possible to make use of predictor–corrector methods such as the Kalman filter [5.17, 18] or sequential importance sampling [5.19], provided appropriate statistical characterizations of the sensing system are available. The data association problem, if present, can be addressed using a variety of general techniques such as EM [5.12] or more specialized modifications to the previously cited predictor–corrector methods [5.20].

It is often the case that sensor data is corrupted by occasional nonsensical values. For example, the laser range finder in our example may occasionally return a spurious range value due to a reflection. Many commonly used estimation techniques are not robust to such so-called data outliers. Techniques from robust statistics [5.21] can be used to improve the performance of sensing and estimation systems in such cases.

Finally, we may want to consider what information is actually important for the task at hand. Most of the techniques above presume that the goal is to produce an accurate estimate of a set of continuous parameters closely related to the underlying data. However, in some tasks, the parameter values themselves may not be what is of interest. For example, suppose that the goal of our robot is to drive through a doorway. Although this clearly depends on an ability to estimate the width of the door (a continuous parameter), the decision is ultimately binary. This problem can be codified as a *decision problem*. Decision problems can be modeled using concepts from *decision theory* [5.22] including zero–one loss functions, likelihood ratios, or probability ratios. For example, in the case of fitting through a door, for a low-priority task there may be a low cost associated with not attempting to move through this particular door (necessitating replanning to find an alternative route) relative to attempting to navigate through an opening that is too small (risking damage to the robot or the door, or both). Conversely, if the task is urgent, more risky behavior may be warranted.

For any given task (or decision), the amount of information necessary to reach the decision may vary, for example, if the doorway is quite wide, it may require relatively little information to safely navigate through it. Conversely, a tight fit may require close inspection before a decision can be reached. The problem of determining the type and/or amount of information necessary to reach a decision is referred to variously as the *sequential sampling problem* [5.22], the sensor control problem, or the sensor planning problem [5.23–25].

5.4.1 Point Estimation

In our robot localization example we saw several cases where the key problem was to estimate an unknown quantity that can be represented as a point in a vector space. Examples include the location of a two-dimensional (2-D) or three-dimensional (3-D) point or the location of a robot. We also saw examples where the problem was to locate the pose (position and orientation) of the robot, or parameters of a line segment. The latter differ in that the underlying parameter space is not a vector space. This introduces some additional unique problems. We refer the reader to [5.26, 27] for further discussion, and in the remainder of this chapter restrict our attention to *point estimation* problems on vector spaces. In our discussion, we assume the reader is familiar with multivariate Gaussian distributions as described in [5.28] and basic linear algebra [5.29].

In the remainder of this section, we consider the following general problem.

Given: an observation model

$$y = f(x, \eta)\ . \tag{5.2}$$

Estimate: $x \in \mathrm{Re}(n)$ from observations $y \in \mathrm{Re}(m)$ where η is an unknown disturbance taking values in $\mathrm{Re}(k)$ and f is a known mapping from $\mathrm{Re}(k+n)$ to $\mathrm{Re}(m)$.

We divide our discussion into two topical areas:

- Methods for performing estimation on batch and sequential data when f is linear,
- Methods for performing estimation on sequential data when f is nonlinear.

Estimation Techniques for Batch and Sequential Data with Linear Models

In this section, we discuss linear and linearized estimation techniques for sequential data, including the Kalman filter and extensions thereof. Our goal is to provide an overview of techniques available. The reader may also wish to consult more in-depth references such as [5.18, 30, 31] and (Chap. 35) for additional information.

We first consider the case when f in (5.2) is linear in its arguments. In this case, we can write

$$y = \mathbf{F}x + \mathbf{B}\eta\ , \tag{5.3}$$

where $\mathbf{F} \in \mathrm{Re}(m \times n)$ defines the (linear) relationship between the unknown x and the observation y and $\mathbf{B} \in \mathrm{Re}(m \times m)$. For the moment, we will drop $\mathbf{B}$ and assume that η represents the complete disturbance model of the system.

The *least-squares* method of estimating x from y proceeds by solving the optimization problem

$$\min_x \|\mathbf{F}x - y\|^2\ . \tag{5.4}$$

This optimization has a unique solution $\hat{x}$ if and only if the matrix $\mathbf{F}$ has full column rank. In this case, the solution can be computed by solving the following linear system

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\hat{x} = \mathbf{F}^{\mathrm{T}}y\ . \tag{5.5}$$

In some cases, there may be reason to believe that some observed elements are more reliable than others, and hence should contribute more to the final estimate. This information can be incorporated by modifying (5.4) to include a diagonal positive-definite weighting matrix $\mathbf{W}$ as

$$\min_x (\mathbf{F}x - y)^{\mathrm{T}}\mathbf{W}(\mathbf{F}x - y)\ . \tag{5.6}$$

The solution is then given by solving

$$(\mathbf{F}^{\mathrm{T}}\mathbf{W}\mathbf{F})\hat{x} = \mathbf{F}^{\mathrm{T}}\mathbf{W}y\ . \tag{5.7}$$

Although (5.3) included a disturbance component (in the form of η), the parameter estimates computed in (5.5) or (5.7) made no explicit use of this quantity. However, we can often model the noise characteristics of the underlying sensor using a statistical model and recast our original estimation problem to incorporate this information. One common method is to compute the *maximum-likelihood estimate* (MLE), which is a value $\hat{x}$ such that

$$p(y|\hat{x}) = \max_x p(y|x)\ . \tag{5.8}$$

For the linear additive model of (5.3), the likelihood function can be expressed in a particularly simple form. Suppose that η is described by a fixed, known probability density function D. The likelihood function is then given by

$$p(y|x) = D(y - \mathbf{F}x)\ . \tag{5.9}$$

The MLE can be related to the previous least-squares method as follows. Suppose that $\eta \sim N(0, \mathbf{\Lambda})$, where N denotes a multivariate Gaussian density function with (mean) 0 and covariance $\mathbf{\Lambda}$. Upon observing that the maximizing the value of the likelihood function is equivalent to minimizing the negative log of the likelihood function, a short series of calculations shows that the optimal maximum-likelihood estimate is computed by weighted least squares with $\mathbf{W} = \mathbf{\Lambda}^{-1}$.

Finally, there is often a reason to include the idea that some parameters are more likely a priori to occur as others. For example, when observing a car driving on an expressway, a velocity of 60 mph is much more likely than either 20 or 300 mph. This information can be captured in *prior statistics* on the unknown value x.

Given a prior probably density on x, $p(x)$, *Bayes theorem* states that

$$p(x|y)=\frac{p(y|x)p(x)}{p(y)}=\frac{p(y|x)p(x)}{\int p(y|x)p(x)\,\mathrm{d}x}\;. \tag{5.10}$$

The *maximum a posteriori probability* (MAP) estimate is the value $\hat{x}$ such that

$$p(\hat{x}|y)=\max_x p(x|y)\;. \tag{5.11}$$

In general, the solution to this optimization problem can be quite complex. Rather than pursue this course further, we consider another alternative. Namely, provided the second moments of $p(x|y)$ exist, it is possible to produce a *least-squares* estimate, in a statistical sense, by solving the following optimization problem over an unknown function δ

$$\min_\delta E\;\|\delta(y)-x\|^2\;. \tag{5.12}$$

That is, the best function δ is one that produces an estimate of x from y with *minimum mean-square error* (MMSE). Thus, the estimator δ is often referred to as an MMSE estimator.

It can be shown that, in the general case, the *optimal* decision rule δ^* is the conditional mean [5.22]

$$\delta^*(y)=E\,[x\mid y]\;. \tag{5.13}$$

Unfortunately, this expression, as with the MAP estimate defined above, can be extremely difficult to compute for the general case. Later we consider methods for computing approximations to (5.13). For now, we again consider our previous linear observation model (5.3) (without B). Additionally, we suppose that x and η are *independent* random variables with finite second moments, and both are *zero-mean* random variables. Note that the latter is not really a restriction since it can be accomplished by simply defining a new variable $x'=x-E[x]$. Finally, we will consider only *linear* functions δ, that is, we can write $\hat{x}=\delta(y)=\mathbf{K}y$.

With this, (5.12) can be expanded as

$$\begin{aligned}E\;\|\delta(y)-x\|^2&=E\;\|\mathbf{K}y-x\|^2\\&=E\;\|\mathbf{K}(\mathbf{F}x+\eta)-x\|^2\\&=E\;\|(\mathbf{KF}-I)x\|^2+E\;\|\mathbf{K}\eta\|^2\\&=\operatorname{tr}\left[(\mathbf{KF}-I)\mathbf{\Lambda}(\mathbf{KF}-I)^{\mathrm{T}}+\mathbf{K\Sigma K}^{\mathrm{T}}\right].\end{aligned} \tag{5.14}$$

Here, the independence of x and η and the fact that they are both zero mean has eliminated several terms. The final step makes use of the fact that $\|x\|^2=\operatorname{tr}(xx^{\mathrm{T}})$.

Taking derivatives with respect to $\mathbf{K}$ and setting them equal to zero yields the solution

$$\mathbf{K}=\mathbf{\Lambda F}^{\mathrm{T}}(\mathbf{F\Lambda F}^{\mathrm{T}}+\mathbf{\Sigma})^{-1}\;. \tag{5.15}$$

Thus, in this case the optimal estimate is given by a linear function of the observation, where the linear term depends only on the variance of the underlying random variables and the linear term defining the observation system.

If x is not zero-mean, but has mean μ, it is not hard to show that the optimal estimate is

$$\hat{x}=\mathbf{K}y+(\mathbf{I}-\mathbf{KF})\mu\;, \tag{5.16}$$

and that the variance of the estimate Λ^+ is

$$\mathbf{\Lambda}^+=(\mathbf{I}-\mathbf{KF})\,\mathbf{\Lambda}\;. \tag{5.17}$$

The interested reader may wish to work this out for a few simple cases, for example, if $\mathbf{\Lambda}=\mathbf{\Sigma}$ and $\mathbf{F}=\mathbf{I}$, $\mathbf{K}=1/2\mathbf{I}$ and thus $\hat{x}=y+\mu$ – a simple average – with variance $\mathbf{\Lambda}^+=1/2\mathbf{\Lambda}$.

When both the observation noise and prior statistics are Gaussian distributions, then it can be shown that the solution we have derived is also the MAP estimate for the unknown x [5.22].

The Kalman Filter

With this as background, we are now in a position to define the discrete-time Kalman–Bucy filter [5.32] for linear systems. Consider the following time-series model

$$x_{t+1}=\mathbf{G}x_t+w_t\;, \tag{5.18}$$

$$y_t=\mathbf{F}x_t+\eta_t\;, \tag{5.19}$$

where $\mathbf{G}$ is an $n\times n$ matrix describing the system time evolution and x_0 is distributed according to a Gaussian distribution with mean $\hat{x}_0$ and variance $\mathbf{\Lambda}_0$. In addition w_t and η_t are zero-mean Gaussian independent random variables for all t, w_t is independent of $w_{t'}$ for all $t\neq t'$, and likewise η_t is independent of $\eta_{t'}$ for all $t\neq t'$. Finally, η_t has variance $\mathbf{\Sigma}_t$ and w_t has variance $\mathbf{\Omega}_t$.

Given an observation y_1 it is possible, using the derivation of the previous section, to compute an updated estimate $\hat{x}_1$ with variance $\mathbf{\Lambda}_1$. Note, that the solution is a linear combination of two Gaussian random variables: the observation value y_1 and the prior estimate $\hat{x}_0$. As any linear combination of Gaussian random variables is also a Gaussian random variable, it follows that the updated estimate is also Gaussian.

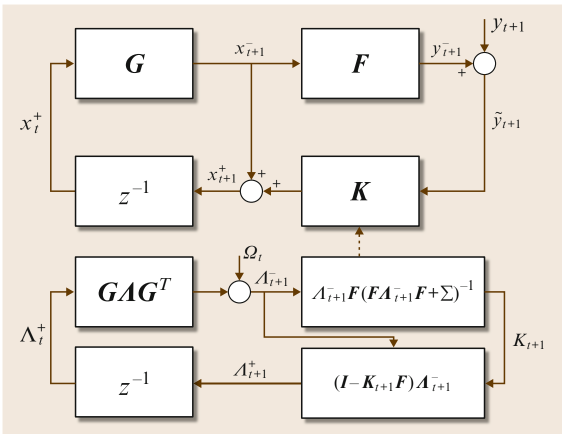


Fig. 5.13 A summary of the Kalman filter

Now, we add one additional step: projection through the dynamics model. To describe this, superscripts minus and plus will denote before and after the estimation step, respectively. Thus, given an estimate $\hat{x}_t^+$ with variance $\mathbf{\Lambda}_t^+$, the projection ahead one time step produces

$$\hat{x}^-_{t+1} = \mathbf{G}x_t^+ \, , \tag{5.20}$$

$$\mathbf{\Lambda}^-_{t+1} = \mathbf{G}\mathbf{\Lambda}_t^+\mathbf{G}^\mathrm{T} + \Omega_t \, . \tag{5.21}$$

At this point, a new observation y_{t+1} is acquired and the cycle repeats. The summarization of the complete Kalman filtering algorithm for linear systems is shown in Fig. 5.13.

It is possible to show that the Kalman filter is the *optimal* filter, under the stated assumptions, in the mean-square sense. It is also the optimal *linear* filter when either or both Gaussian assumptions do not hold.

Nonlinear Estimation Techniques for Sequential Data

The results of the previous subsection presume a linear form for the relationship between the observation and system state, additive noise, and a linear relationship describing the state evolution. Furthermore, the stated results are globally optimal for systems with Gaussian observation and driving noise, but are only the best *linear* estimator if the noise sources are non-Gaussian.

As noted at the outset, the more general nonlinear (discrete-time) system description is

$$\begin{aligned} x_{t+1} &= g_t(x_t) + w_t \, , \\ y_t &= f_t(x_t) + \eta_t \, , \end{aligned} \tag{5.22}$$

where, for the moment, the noise model continues to be additive.

Although this model contains nonlinear elements, it is still possible to apply a variant of the Kalman filter, the extended Kalman filter (EKF) by making use of the Taylor-series expansion of the nonlinear elements about the current estimates. Let $\mathbf{J}_f$ (resp. $\mathbf{J}_g$) denote the Jacobian matrix of the function f (resp. g). Supposing that an estimate at time step $t-1$ exists, the first-order expansion of (5.22) about this point yields

$$x_{t+1} = g_t(\hat{x}_{t-1}) + \mathbf{J}_g(\hat{x}_{t-1})(x_t - \hat{x}_{t+1}) + w_t \, , \tag{5.23}$$

$$y_t = f_t(\hat{x}_{-1}) + \mathbf{J}_f(\hat{x}_{t-1})(x_t - \hat{x}_{t-1}) + \eta_t \, . \tag{5.24}$$

Rearranging yields a linear form appropriate for the previously defined Kalman filter

$$\tilde{x}_{t+1} = x_{t+1} - g_t(\hat{x}_{t-1}) + \mathbf{J}_{g_t}\hat{x}_{t-1} = \mathbf{J}_{g_t}x_t + w_t \, , \tag{5.25}$$

$$\tilde{y}_t = y_t - f_t(\hat{x}_{t-1}) + \mathbf{J}_{f_t}\hat{x}_{t-1} = \mathbf{J}_{f_t}x_t + \eta_t \, . \tag{5.26}$$

In this form, $\tilde{x}$ and $\tilde{y}$ are new *synthetic* state and observation variables, $\mathbf{J}_g(\hat{x}_{t-1})$ plays the role of G, and $\mathbf{J}_f(\hat{x}_{t-1})$ plays the role of F.

It is worth noting that the EKF iterations are essentially a form of weighted Newton iterations (i. e., an iterative nonlinear estimation method). As a result, it is often useful to iterate more than once *on the same observation* while holding the variance terms fixed. This allows the estimator to converge to a solution in the presence of large disturbances or significant nonlinearities. Only after convergence are the variance terms updated. This version of the Kalman filter is referred to as the *iterated extended Kalman filter* (IEKF).

5.4.2 Other Approaches to Estimation

In the previous section, we reviewed a common and widely used estimation method. However, there are several alternative methodologies for solving parameter estimation problems. Here we briefly introduce two: sequential importance sampling and graphical models.

Sequential Importance Sampling

Much of the discussion heretofore has centered around the notion of approximating everything known about the system state using an estimated mean and covariance. An alternative presents itself by simply going back to Bayes theorem which states, in general, that

$$p(x_n|y_1, y_2 \dots y_n) = \frac{p(y_1, y_2 \dots y_n|x_n)p(x_n)}{p(y_1, y_2 \dots y_n)} \, . \tag{5.27}$$

Assuming that y_n is independent of all prior observations and states given x_n, and that x_n is independent of x_{n-k} for $k > 1$ given x_{n-1}, this expression simplifies to

$$p(x_n|x_{n-1}, y_n) = \frac{p(y_n|x_n)p(x_n \mid x_{n-1})}{p(y_n \mid x_{n-1})} . \tag{5.28}$$

Recall that the optimal mean-square estimate is given by the conditional mean which, in this case, is

$$\delta^*(y_n) = E\left[x_n \mid y_n\right] . \tag{5.29}$$

In fact, we essentially showed that the Kalman filter is a special case of this result for linear systems corrupted by Gaussian noise.

The difficulty in implementing this procedure in the general case ultimately comes down to the problem of representing and computing with the distributions that arise in the nonlinear, non-Gaussian case. However, suppose that the heretofore continuous variable x_n only took on a discrete set of values. In this case, computing Bayes theorem and other associated statistical quantities reduces to a straightforward set of computations on this discrete set of variables. This can be simply done for any distribution and any set of transformations.

Sequential important sampling (also known as particle filtering, condensation, and a variety of other names) is a way of applying this approach to continuous variables in a statistically sound manner. In order to perform sequential importance sampling, it is assumed that:

1. It is possible to *sample from* the likelihood function $P(y_n \mid x_n)$, and
2. It is possible to *sample from* the dynamical model $P(x_n \mid x_{n-1})$.

Note the emphasis on sampling – there is no need to explicitly exhibit an analytical form of the likelihood function or of the dynamical model.

Given this, sequential important sampling, in its simplest form, can be written as follows:

1. Let $\pi_{n-1} = \{\langle x^k_{n-1}, w^k_{n-1}\rangle, k = 1,2,\dots N\}$ represent a set of sample points x^k_{n-1} together with a set of weights w^k_{n-1} with $\sum w^k_{n-1} = 1$.
2. Compute a new set of N samples $\pi^-_{n-1} = \{\langle x^k_n, 1/N\rangle, k = 1,2,\dots N\}$ as:
   a) Choose a sample point x^{k-1}_{n-1} with probability proportional to its weight w^{k-1};
   b) Sample from $P(x_n \mid x^k_{n-1})$ given x^k_n with weight $1/N$;
3. Compute $\pi_n = \{\langle x^k_n, P(y_n \mid x^k_n)\rangle, k = 1,2,\dots N\}$.

It is easy to see that this set of steps is now in the form of a recursive filter. Furthermore, at any time any statistic of the associated distribution can be approximated from the set of samples and associated weights.

Sampling-based filters of this form have found wide applicability in a variety of challenging areas where linear estimation techniques do not suffice. These techniques have been particularly successful, for problems with low state dimension (typically $n \le 3$) and well-constrained dynamics. For higher-dimensional problems or systems exhibiting high dynamic variability, the number of particles necessary to obtain good approximations can become prohibitively large. However, even in these cases, sampling-based systems can sometimes be engineered to produce acceptably good results.

Graphical Models

Graphical models are a class of models that represent dependence and independence relationships among a set of variables. Common examples of graphical models include Bayes nets, influence diagrams, and neural nets. Graphical models are quite general – indeed much of this chapter could have been written by first defining graphical models, and exploring specializations that lead to the Kalman Filter, for example. Here, for reasons of space, we focus on Bayes nets as a specific example of graphical models.

A Bayesian network is a directed acyclic graph consisting of nodes representing random variables, and directed arcs representing probabilistic relationships between pairs of random variables. Let parents(X) denote the set of nodes which have arcs terminating at X, and let $X_1, X_2, \dots, X_N$ be the N random variables in the graph. Then we can write

$$P(X_1, X_2, \dots, X_N) = \prod_{i=1}^{N} P(X_i \mid \text{parents}(X_i)) . \tag{5.30}$$

For example, a Bayesian network representing a mobile robot performing localization is shown in Fig. 5.14. This graphical model encodes the sequential form of the problem and is thus an example of a so-called *recurrent*

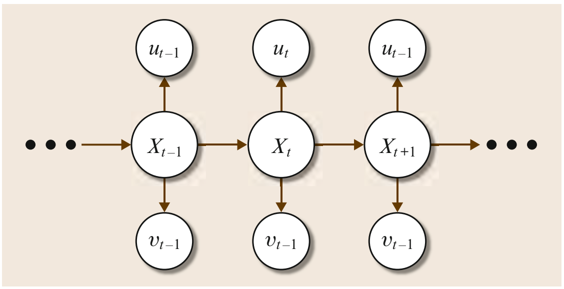


Fig. 5.14 An example of robot localization expressed as a graphical model

network. More discussion of such models can be found in [5.33].

The structure of a Bayesian network encodes various independence relationships among variables. By exploiting these independence relationships, it is possible to design efficient inference algorithms. In particular, graphs which are acyclic even in their undirected form (referred to as *polytrees*) admit linear time inference algorithms. More general graphs can be solved using various types of iterative methods. In particular, if the distributions in the network are of a continuous type, variations on sequential importance sampling can be used to solve problems in an approximate sense [5.34].

Conditional Random Fields

In many cases of interest, including many of the examples in this chapter, the end goal is to infer or predict a value or label from observed data. We might then frame the problem by exploring the joint distribution $P(X, Y)$ where X represents some data that is *observed* and Y is what we would like to infer. Recall that

$$P(X, Y) = P(Y|X)P(X) .$$

If we take $X = x$ for some observed values x, then we see that $P(X)$ becomes constant, and inferring a value for Y depends only on $P(Y|X)$. If we were to apply a Bayes Net to this problem, the model would represent the complete joint probability distribution on X and Y, what is referred to as a *generative model.* But, if we know X is always observed, then much of this structure is irrelevant to our problem – we don't care about the probabilistic structure of X. This observation has given rise to a specialization of graphical models referred to as *Conditional Random Fields*, or CRFs for short.

The immediate value of CRFs is their economy and expressivity compared to graphical models. This has immediate positive implications for the complexity of both learning and inference. Traditionally CRF models are learned using maximum likelihood-based methods using gradient descent or other unconstrained optimization techniques. However, recent methods like Cutting Planes [5.35] and Block Coordinate Frank Wolfe [5.36] pose it as a constrained optimization problem in the form of a Structural Support Vector Machine. These techniques tend to be more computationally efficient and are often more accurate.

CRFs have proven to be very general, and are now extremely widely used for image processing, natural language processing, video processing – nearly any problem where there is a series of data elements

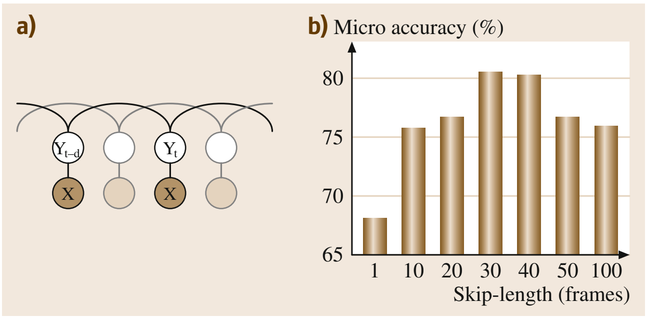


Fig. 5.15 **(a)** A skip chain CRF for inferring symbolic labels (Y) from robot kinematic and video data (X) acquired while a user performed a surgical training task. **(b)** The change in classification accuracy as a function of the skip length (images courtesy of Colin Lea)

from which some prediction is to be performed. For example, Fig. 5.15 shows the graphical structure of a skip-chain CRF designed to compute gesture labels from kinematic and video data acquired from a surgical robot [5.37]. This can be viewed as a discriminative generalization of a Hidden Markov Model (a generative model) that is designed to capture dependencies over a specified period of time (the *skip*). The right side shows the change in labeling performance as a function of the skip length which is now a *tunable* parameter of the model.

An in-depth discussion of CRFs goes well beyond this chapter. The interested reader is referred to [5.38, 39] to learn more about the underlying theory and application of CRFs. Because of their high interest, there are a number of open-source packages for developing and applying CRFs including PyStruct [5.40] for Python and CRF++ for C++.

5.4.3 Robust Estimation Methods

In our previous discussions, we generally assumed that all of the data was *good*, meaning that it was perhaps corrupted by noise but ultimately carried information about the problem at hand. However, in many cases, the data may contain so-called *outliers* – data points that are either much more highly corrupted than typical data, or which are completely spurious. For example, in our mapping application we might occasionally obtain range data through multiple reflections. Thus, while scanning a straight wall, most of the points would lie on a straight line, but occasionally we would have a data point that has a completely inconsistent range value.

Many common estimation methods are quite sensitive to data outliers. Consider a very simple case: estimating a single scalar value x by averaging a series of observations $X_1, X_2, \ldots X_N$. Then we can write our estimate $\hat{x}$ as

$$\hat{x} = \sum_{i=1}^{N} X_i / N \,. \tag{5.31}$$

Now, without loss of generality, suppose that X_N is an outlier. We can rewrite the above as

$$\hat{x} = \sum_{i=1}^{N-1} X_i / n + X_N / n \,. \tag{5.32}$$

It is now easy to see that we can produce *any* value of $\hat{x}$ by manipulating X_n. In short, a single outlier can create an arbitrarily poor estimate. More generally, the solution to any least-squares problem, e.g., estimating a line from laser range data, takes the general form $\hat{x} = \mathbf{M}y$. By the same argument as above, it is easy to show that any least-squares solution is likewise susceptible to outliers.

The field of *robust statistics* studies the problem of estimation or decision making when the underlying data are contaminated by outliers. In robust statistics, there are two important concepts: the *breakdown point* and *influence function*. The breakdown point is the proportion of outliers (i. e., data with arbitrarily large errors) that an estimator can tolerate before producing arbitrarily large errors in an estimate. We argued above that least-squares methods have a breakdown point of 0% since the estimate can be perturbed arbitrarily far by a single observation. By comparison, we might compute an estimate by taking the median of the data, which has a breakdown point of 50% – up to half of the data can be outliers and meaningful results may still be produced.

Whereas the breakdown point quantifies *how many* outliers can be tolerated, the influence function quantifies *how much* an outlier affects an estimate. In the case of least squares, the influence function is linear. One way of creating new estimators with better robustness is the method of *M-estimators* [5.21]. To produce an M-estimate, we consider the following minimization problem

$$\min_{\hat{x}} \sum_{i=1}^{N} \rho(\hat{x}, y_i) \,. \tag{5.33}$$

Note that defining $\rho(a,b) = (a-b)^2$ leads to a least-squares solution. However, we can now choose other functions with better resistance to outliers. Fig. 5.16 shows three common examples.

Note that, in general, the optimization of (5.33) is nonlinear and the result will often not exist in closed form. Interestingly, it is often possible to solve this problem using the method of *iteratively reweighted least squares* (IRLS) [5.30, 41]. The idea behind IRLS is quite simple. Recall that in (5.7) we introduced a weighting matrix $\mathbf{W}$. Suppose that, through some means, we knew which data points were outliers. In this case, we could simply set the weights for those points to zero, and the result would be the least-squares estimate on the remaining (good) data.

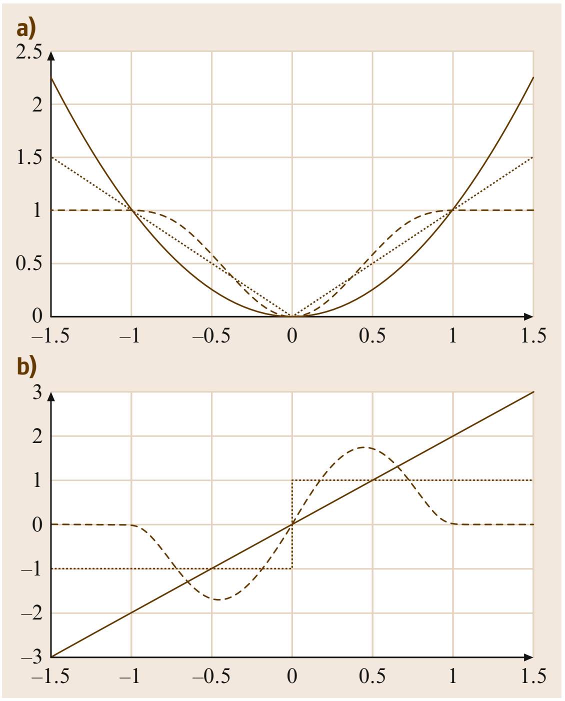


Fig. 5.16 **(a)** Three common robust M-estimation functions: the square function, the absolute value, and the Tukey biweight function. **(b)** The corresponding influence functions

In IRLS, we alternate between hypothesizing outliers (through reweighting) and solving to produce a solution (through least squares). Typically, the weight for a point depends on the residual error of the estimate. That is, suppose we compute

$$r = y - \mathbf{F}\hat{x} \,. \tag{5.34}$$

Let $\psi(y) = \mathrm{d}\rho / \mathrm{d}x \,|_{\hat{x}}$; then we can set $\mathbf{W}_{i,i} = \psi(y)/r_i$. It can be shown that in many cases this form of weighting will lead to convergence. An example of using IRLS techniques for video tracking is shown in Fig. 5.17.

Voting-Based Methods

Another common method for dealing with outliers is to choose a set of data and let it *vote* for a result. We dis-

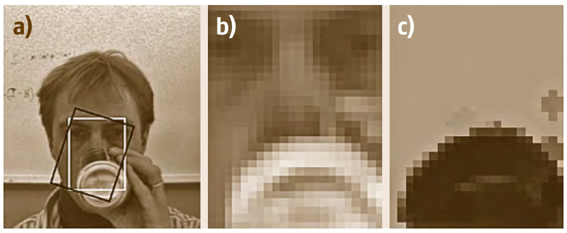


Fig.5.17a–c An example of using an M-estimate implemented via IRLS for visual tracking (after [5.42]). **(a)** Results of a face tracker in a single frame of video. The black frame corresponds to a tracking algorithm without outlier rejection and the white frame corresponds to the algorithm with outlier rejection. **(b)** Magnified view of the region in the white frame; **(c)** the corresponding weighting matrix in which darker areas mark outliers

cuss two common methods: RANSAC [5.11] and least median of squares (LMedS) [5.43].

In both cases, we start with the idea that, amidst all of the data (including outliers), there is an estimate that is consistent with the *good* data. The problem is to choose that estimate. Consider, however, our problem of estimating a line from laser data, and suppose we have 100 laser points. All we really need is to choose two points correctly, fit a line, and then count how many other points are consistent with this line. If we (conservatively) estimate that $3/4$ of the data is good, then the odds of choosing two *good* points is $9/16$, or equivalently, the odds of one or both points being outliers is $7/16$. If we now repeat this process a few (e.g., ten) times, then the odds that *all* of our choices are bad is $(7/16)^{10} = 0.025\%$. To put it in other terms, there is a 99.975% chance we have chosen a *good* pair of points.

How do we decide to accept a sample? In RANSAC, we *vote* by counting the number of samples that are consistent with an estimate to within a given distance threshold. For example, we would choose points that are within a fixed distance to the line we estimated. We choose the candidate estimate with the largest number of votes. In LMedS, we instead compute the median distance of all of the samples to the line. We then choose the estimate with the least median value.

It is not hard to see that LMedS has a breakdown point of 50% of the data. RANSAC, on the other hand, can have a breakdown point that is potentially larger, but it requires the choice of a threshold. RANSAC also has the advantage that, once the inliers are identified, it is possible to compute a least-squares estimate from them, thus reducing the noise in the estimate.

Both RANSAC and LMedS can also provide good starting solutions for a robust iterative method such as IRLS.

5.4.4 Data Association Techniques

The previous section considered the case where there is a *known* relationship between observations and a quantity to be estimated. However, as was illustrated in our initial mobile robot mapping problem, it may be the case that we also have to compute this correspondence in conjunction with estimation. In this case, an essential step in estimation is the *data association* problem: producing a correspondence between the observed data and quantities to be estimated.

The literature on this problem is enormous; here we will focus on a few specific methods that have found wide use. We will also separate our discussion into *causal* (or sequential) association methods commonly used when filtering time-series data and *noncausal* (or batch) methods that can be used when the complete data set is available for processing. The latter is typically treated with methods for data *clustering*.

In both cases, we can extend our previous models and notation to include uncertainty as to the underlying source of the data. To this end, we will use a superscript on quantities to denote the observation model. Thus, our observation model becomes

$$x^k_{t+1} = g^k(x^k_t) + w^k_t \,, \tag{5.35}$$

$$y^k_t = f^k_t(x^k_t) + \eta^k_t \,, \tag{5.36}$$

where $k = 1 \ldots M$.

Clustering on Batch Data

Following the same course as our previous discussion on point estimation, let us first consider the case where we do not make any statistical assumptions about the data, and we have no system dynamics. Thus, we are simply given the observations $y_1, y_2, \ldots, y_M$. We have unknown underlying parameters $x^1, x^2, \ldots, x^N$ (for the moment, we take N as known). Our goal it to compute an *association mapping* π such that $\pi(j) = k$ if and only if y_j arose from the model parameters x^k.

k-Means Clustering

The k-means algorithm for clustering and data association is simple, well established, and forms a good starting point for our discussion. Here, we assume that $f(x) = x$ – that is, we are provided with noisy observations of the underlying state vectors. The k-means algorithm then proceeds as follows:

1. Pick N cluster centers $\{\hat{x}^i\}$.
2. For each observation y_j, associate it with the closest cluster center, that is, set $\pi(j) = i$, where

$$d(\hat{x}^i, y_j) = \min_k d(\hat{x}^k, y_j) \tag{5.37}$$

for some distance function d (typically the Euclidean distance).

3. Estimate the mean of the observation associated with each cluster center as
$$\hat{x}^i = \sum_{j,\pi(j)=i} y_j \, . \tag{5.38}$$
4. Repeat steps 2 and 3.

In many cases and with good initialization, k-means works quite well. However, it can also fail to produce good clusters, and there is no guarantee that it will even converge to a solution. It is common to repeat the algorithm several times from different initial conditions and take the result that has the *best* outcome. Note also that the extension to linear observation models is straightforward by including $\mathbf{F}$ in (5.3) by defining
$$d(\hat{x}^i, y_j) = \|\mathbf{F}\hat{x}^i - y_j\| \tag{5.39}$$
and replacing (5.38) with the corresponding least-squares estimator. Going a step further, if we have a statistical model for observed data, then we could make use of the likelihood function introduced earlier and define $d(\hat{x}^i, y_j) = p(y_j|\hat{x}^i)$ and make use of the MLE in (5.38).

One disadvantage of the k-means algorithm is that, even when we have known statistical models, it is not guaranteed to converge. However, a variation, known as *expectation maximization*, can be shown to converge.

Expectation Maximization for Data Association and Modeling

The expectation-maximization (EM) algorithm [5.44] is a general statistical technique for dealing with missing data. In previous discussion, we made use of maximum-likelihood estimation to maximize the conditional probability of observed data given a set of unknown parameters. However, our use of MLE presumed that we had complete knowledge of the data. In particular, we knew the association between the data elements and models.

Let use now assume that some of our data is missing. To this end, define $\mathcal{Y}_\mathrm{O}$ and $\mathcal{Y}_\mathrm{U}$ as the *observed* and *unobserved* data, respectively. We then note that we can write
$$p(\mathcal{Y}_\mathrm{O}, \mathcal{Y}_\mathrm{U}|x) = p(\mathcal{Y}_\mathrm{U}|\mathcal{Y}_\mathrm{O}, x)p(\mathcal{Y}_\mathrm{O}|x) \, . \tag{5.40}$$

Suppose now that we make a guess for $\hat{x}$, and we have a distribution over the unknown data $\mathcal{Y}_\mathrm{U}$ (where this comes from we will discuss in a minute). It follows that we could compute the *expected value* of the log-likelihood function (recall that maximizing the log likelihood is equivalent to maximizing the likelihood) as
$$Q(x,\hat{x}) = E_{\mathcal{Y}_\mathrm{U}}\left[\log p(\mathcal{Y}_\mathrm{O}, \mathcal{Y}_\mathrm{U}|x)|\mathcal{Y}_\mathrm{O}, \hat{x}\right] \, . \tag{5.41}$$
Note that we differentiate between the fixed value $\hat{x}$ that is usually needed to define the distribution over the unknown data and the unknown x of the log-likelihood function.

Ideally, we would then like to choose values for x that make Q large. Thus, we can choose a new value according to the iterative rule
$$\hat{x}_i = \arg\max_x Q(x, \hat{x}_{i-1}) \, . \tag{5.42}$$
What can be shown is that this iteration will converge to some local maximum of the objective function Q. It is important to note that there is no guarantee that this is, however, the *global* maximum.

How do we connect this with clustering? We consider the observed data to be just that, the data we have observed. Let the unobserved data be the *association values* $\pi(j)$, $j = 1, 2, \ldots M$ that determine which model the observed data items originate from. Note that this is a discrete random variable. Let us further assume that N underlying clusters are distributed according to a Gaussian distribution with mean x_i and covariance Λ_i. Let the unconditional probability that a particular data item y_j comes from cluster i be α_i. The unknown parameters are then $\theta = \{x_1, x_2, \ldots, x_N, \Lambda_1, \Lambda_2, \ldots, \Lambda_N, \alpha_1, \alpha_2, \ldots, \alpha_N\}$. We now use $^-$ and $^+$ to denote prior and updated parameter estimates, respectively. For conciseness, we also define $w_{i,j} = p(\pi_j = i|y_j, \theta)$ and we use a superscript $^+$ to denote updated parameter estimates. Then, after a series of calculations [5.44], the EM algorithm for data clustering becomes

E-Step:
$$w_{i,j} = \frac{p(y_j|\pi(j) = i, \theta)\alpha_i}{\sum_i p(y_j|\pi(j) = i, \theta)\alpha_i} \, . \tag{5.43}$$
M-Step:
$$\hat{x}_i^+ = \frac{\sum_j y_j w_{i,j}}{\sum_j w_{i,j}} \, , \tag{5.44}$$
$$\Lambda_i^+ = \frac{\sum_j y_j (y_j)^t w_{i,j}}{\sum_j w_{i,j}} \, , \tag{5.45}$$
$$\alpha_i^+ = \frac{\sum_j w_{i,j}}{\sum_i \sum_j w_{i,j}} \, . \tag{5.46}$$

From this, we can see that EM produces a type of soft clustering, as opposed to k-means which produces

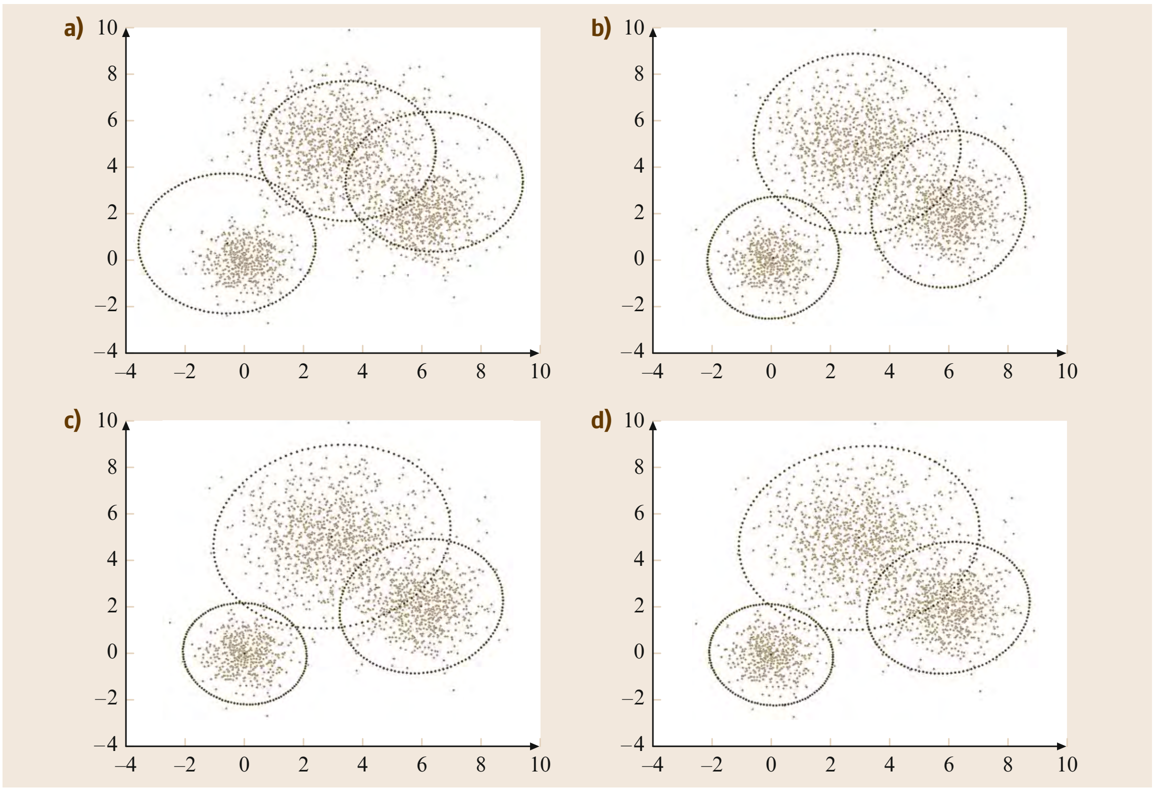


Fig.5.18a–d An example of clustering using expectation maximization. The figures are the results at iterations **(a)** 1, **(b)** 2, **(c)** 5, and **(d)** 10

specific decisions (in terms of $w_{i,j}$) as to which cluster an observation belongs. In fact, the result of estimation is the maximum-likelihood estimate of a *Gaussian mixture model* which has the form

$$p(y|\theta) = \sum_j \alpha_j N(y|\hat{x}_j, \mathbf{\Lambda}_j) , \quad (5.47)$$

where $N(\cdot)$ denotes a Gaussian density function. Fig. 5.18 shows the results of executing the EM algorithm on data sampled from a Gaussian mixture model.

Recursive Filtering

In the batch methods described above, we do not have a priori information on state parameters. In the case of recursive filtering, we have the advantage that prior state estimates, $\hat{x}_t^k$ and Λ_t^k, are available for processing at time $t+1$. As before, for data $y_t^i, i = 1 \dots N$, the problem is to determine a mapping $\pi : \{1 \dots N\} \to \{1 \dots M\}$ which *associates* data element i to model $k = \pi(i)$. In some cases, it is also useful to include an *outlier process* to handle data that comes from no known model. For this purpose, we can include 0 in the range of the function, and use the mapping to zero as an outlier.

Nearest-Neighbor Association

Analogous to k-mean clustering, a simple way of producing a data association is to compute the data association value as

$$\pi(i) = \arg\min_j d(\mathbf{F}^j \hat{x}^j, \hat{y}^i) . \quad (5.48)$$

However, nearest-neighbor methods do not take into account what we know about either the sensor data or the estimate. That is, we may have a very very good estimate of some model i and a very very bad estimate for some other model j. If a sensor observation is equidistance between them, does it make sense to flip a coin? Odds are that it is more likely to come from j (with a larger variance) than i (with a smaller variance).

A commonly used measure that can take this into account is the *Mahalanobis* distance [5.45]. The idea is to weight each value by its variance as

$$m(y_1, y_2) = (y_1 - y_2)(\mathbf{\Lambda}_1 + \mathbf{\Lambda}_2)^{-1}(y_1 - y_2)^{\mathrm{T}} . \quad (5.49)$$

Thus, distances are scaled inversely with uncertainty. In the case above, the observation with a higher variance would produce the smaller distance, as desired.

Even using this as a weighting method, it is still possible that we will make an error in data association. From an estimation point of view, this will introduce an outlier in the estimation process with, as discussed above, potentially disastrous results. Another approach, analogous to IRLS, is instead to weight the data based on the distance to a model. This leads naturally to the notion of a *data association filter*. We refer the reader to [5.20] for extensive discussions of these techniques.

5.4.5 Modeling Sensors

To this point, we have introduced several sensing modalities, and we have discussed several methods for estimation. However, the latter often rely on having statistical models for the former. Thus, no chapter on sensing and estimation would be complete without a short discussion of modeling sensors.

Developing a sensor model potentially involves four major elements:

1. Creating a physical model
2. Determining a sensor calibration
3. Determining an error model
4. Identifying failure conditions.

The physical model is the relationship f between the underlying quantities of interest (x) and the available data (y). In many cases, this relationship is obvious, e.g., the distance from a laser sensor to a surface in the world. In others, it may be less so, e.g., what is the right model relating intensities in multiple camera images to the distance to an observed point? In some cases, it may be necessary to include computational processes, e.g., feature detection and correspondence, in the sensor model.

Once a physical model is determined, there is often a process of sensor calibration. Such procedures are typically specific to the sensor in question, for example, the imaging geometry of a perspective camera system requires identification of two scale parameters (governing image scale) and the location of the optical center (two additional parameters). There are also often lens distortion parameters. These parameters can only be determined by a careful calibration procedure [5.7].

Once a calibrated physical sensor model is available, determining an error model typically involves performing an identification of the statistical parameters. Ideally, the first step is to determine an empirical distribution on errors. However, this can often be difficult, as it requires knowing accurate *ground truth* for the underlying unknown parameters. This often requires the development of a laboratory setup that can simulate the expected sensing situation.

Given such an empirical distribution, there are several important questions, including:

1. Are observations statistically independent?
2. Is the error distribution unimodal?
3. Can the essential aspects of the empirical error be captured using common statistical quantities such as the data variance?

We refer the reader to books on statistics and data modeling [5.46] for further information on this topic.

Finally, it is important to understand when sensors can and cannot provide reliable data, for example, a laser sensor may be less accurate on dark surfaces than on light ones, cameras do not produce meaningful data if the lighting is too bright or too dark, and so forth. In some cases, there are simple clues to these conditions (e.g., simply looking at the intensity histogram of a camera image can quickly determine if conditions are suitable for processing). In some cases it is only possible to detect conditions in context (e.g., two range sensors disagree on the distance to a surface). In some cases failure is only detectable in retrospect, e.g., after a 3-D surface model is built it is apparent that a hypothesized surface would be occluded by another and must therefore be a multiple reflection. In a truly robust sensing system, all available possibilities for verifying sensor operation should be exploited.

5.4.6 Other Uncertainty Management Methods

Due to the limitations of space, we have necessarily limited our discussion to cover the most commonly used sensing and estimation methods. It is important to note that many other alternative uncertainty management methods have been proposed and employed with success.

For example, if it is known that sensing error is bounded, constraint-based methods can be quite effective at performing point estimation [5.47, 48]. Alternatively, if only partial probability models can be identified, Dempster–Shafer methods can be employed to make judgments [5.49].

Fuzzy logic allows graded membership of a set. With fuzzy set theory it is possible to have partial membership. As an example in classification of data it might be difficult to select between two categories such as *average* and *tall* and gradual shifts may make sense. Such methods have for example been used for situation assessment and navigation as reported by [5.50] for the DAMN architecture.

5.5 Representations

Sensor data can be used directly for control but it is also used for estimation of the state of the robot and/or the world. The definition of *state* and the appropriate methods for estimation are closely related to the representation adopted for the application.

There are a rich variety of possible world representations including most typical geometric elements such as points, curves, surfaces, and volumes. A fundamental aspect in robotics is the concept of rigid-body *pose*. The pose of a robot or an entity in the world is characterized by position and orientation with respect to a reference frame.

In general, pose is represented by the pair $(\mathbf{R}, \boldsymbol{H})$. Here $\mathbf{R}$ is the orientation of the object represented by a rotation matrix with respect to a reference frame. Similarly, $\boldsymbol{H}$ represents the translation of the object with respect the reference frame. There is a rich set of potential representations for the transformation between reference frames as detailed in the chapter on Kinematics (Chap. 2) and in [5.51].

Sensory data is acquired in a local sensor reference frame, for example, a sonar transducer, a laser scanner, and a stereo imaging system would all measure distances to surfaces in the world relative to their own frame. However, if the goal is to combine this information into a common world model, the data must be transformed into a robot-centered reference frame, or possibly into a fixed world (inertial) reference frame. In particular, the world-centered reference frame enable simple transfer across robot motions and communication to other robots and/or users.

For the purposes of discussion, most representations for the integration of sensor data can be categorized into four general classes of models:

- Raw sensor data models
- Grid-based models
- Feature-based models
- Symbolic or graphical models.

Naturally, it is also possible to combine elements of these four categories to achieve hybrid models of the environment.

5.5.1 Raw Sensor Representations

For simple feedback control [5.52] it is common to integrate raw sensory data directly into the control system, as in many cases it is unnecessary to have a *world model* for the control. For example, proprioceptive sensing is often used in this manner: basic trajectory control makes direct use of encoder information from joints, and force control operates directly from force or torque information from force sensors.

Raw sensor models are less common with exteroceptive sensing, but there are cases where it can be useful. One example is mobile robot mapping from dense point data. This approach has in particular been made popular for laser range sensors, where scan alignment is used for the generation of point-based world models. The work by [5.53, 54] demonstrates how a number of laser range scans can be combined into a joint model of the environment. More formally a scan of the environment at time t is represented as a point set

$$\mathcal{P}_t = \{p_i = (\rho_i, \theta_i) | i \in 1 \ldots N\} . \tag{5.50}$$

Two different scans $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ are then aligned through a standard rigid body transformation. The estimation of the transformation is typically achieved through use of the ICP algorithm [5.15]: assume that $\boldsymbol{H}^{[0]}$ is an initial estimate of the transformation between the two point sets and that $||\boldsymbol{p}_t - \boldsymbol{p}_{t+1}||$ is the Euclidean distance between a point from $\mathcal{P}_t$ and a point from $\mathcal{P}_{t+1}$. If furthermore CP is a function to locate the closest point from one set in the other set, then let C be the set of point correspondences between the two sets. Through iterations of the following algorithm,

1. Compute $C_k = \cup_{i=1}^{N} \{\boldsymbol{p}_i, CP[\boldsymbol{H}^{[k-1]}(\boldsymbol{p}_i, \mathcal{P}_{t+1})]\}$,
2. Estimate the $\boldsymbol{H}^{[k]}$ that minimizes the LSQ error between the points in C_k until the error has converged

an estimate of the scan alignment can be found and a joint model of the environment can be constructed.

The model is simple to construct and well suited for integration of sensor data from a single modality. Typically the model does not include information about uncertainty and, as the model grows the complexity, $O(\sum_t |\mathcal{P}_t|)$ becomes an issue.

5.5.2 Grid-Based Representations

In a grid-based representation the world is tessellated into a number cells. The cells can contain information about environmental features such as temperature, obstacles, force distribution, etc. The dimensionality of the grid is typically two or three, depending on the application. The tessellation can either be uniform or tree based using quad-tree or oct-trees [5.55]. The tree-based methods are in particular well suited for handling of inhomogeneous and large-scale data sets. In a grid model each cell contains a probability over the parameter set. As an example, when using the grid model for representation of a physical environment, the cell specifies *occupied* (O) or *free* (F) and the cell encodes the probability $P(\text{occupied})$. Initially where there is no information the grid is initialized to $P(\text{O}) = 0.5$ to indicate unknown. It is further assumed that sensor models are available that specify $P(R|S_{ij})$, i. e., the probability of detection objects for a given sensor and location. Using Bayes theorem (5.10) it is now possible to update the grid model according to

$$p_{ij}(t+1) = \frac{P(R|S_{ij}=O)p_{ij}(t)}{P(R|S_{ij}=O)p_{ij}(t)+P(R|S_{ij}=F)(1-p_{ij}(t))} ,$$

where p_{ij} is computed across the grid model whenever new data are acquired.

The grid-based model has been widely used in mobile robotics [5.56, 57] and in medical imaging where image volumes are quite common [5.58]. Volume models can be relative large. As an example a millimeter-resolution grid model of the human head requires 4 GB of storage, and thus demands significant computational resources for maintenance.

5.5.3 Feature Representations

Both the raw sensor representation and the grid-based models contain a minimum of abstraction for the sensory data. In many cases there is an interest in extracting features from the sensor data to reduce the storage requirement and only preserve data that are invariant across motion of the platform or external objects. Features span most standard geometric entities such as points ($\boldsymbol{p}$), lines ($\boldsymbol{l}$), planes ($\boldsymbol{N},\boldsymbol{p}$), curves ($\boldsymbol{p}(s)$), and more general surfaces. For estimation of properties of the external world there is a need for a hybrid model in which collections of features are integrated into a unified model of state.

In general a point is represented in $\mathcal{R}(3)$. Sensors have associated noise and, consequently, in most cases points have an associated uncertainty, typically modeled as Gaussian with mean μ and standard deviation σ. The estimation of the statistics is achieved using first- and second-order moments.

Line features are more difficult to represent. The mathematical line can be represented by the vector pair $(\boldsymbol{p},\boldsymbol{t})$, i. e., a point on the line and the tangent vector. In many practical applications the line has a finite extent, and there is a need to encode the length of the line, which can be achieved using end points, start point, tangent, and length. In some cases it is advantageous

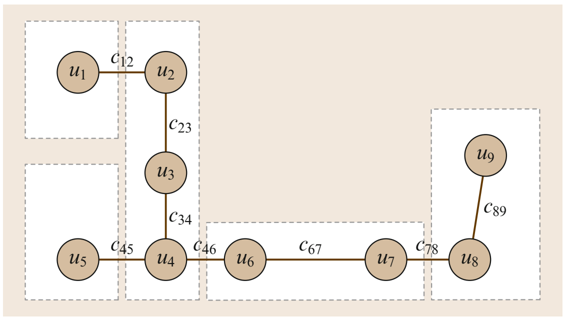


Fig. 5.19 A topological map of a spatial environment

to have a redundant representation of the line model to simplify updating and matching. The relation between end-point uncertainties and other line parameters can be derived analytically, as described in [5.59]. The estimation of line parameters is often based on the previously describe RANSAC method through the use of the *Hough* transform [5.10], which is another voting-based method.

For more complex feature models such as curves or surfaces there is a corresponding need to utilize detection methods that facilitate robust segmentation of features, and estimation of the associated uncertainty. A comprehensive description of such methods is available from [5.44].

5.5.4 Symbolic/Graph-Based Models

All of the representations presented in Sects. 5.5.1–5.5.3 are parametric in nature with limited associated semantics. Methods for the recognition of structures, spaces, locations, and objects have seen major recent progress in particular due to advances in statistical learning theory [5.12, 60]. Consequently, today there exist a variety of methods for the recognition of complex structures in sensor data, such as landmarks, road surfaces, body structures, etc. Given the availability of recognized structures it is possible to represent the environment using the previously discussed graphical models. In general a graph is composed of a set of nodes N and a set of edges E that connect nodes. Both nodes and edges can have attributes associated such as labels and distances. One example of a graph structure is a topological map of the environment as shown in Fig. 5.19. The graph representation could also be a semantic model of the environment (objects and places) or a representation of the composition of an object to assembled.

In terms of model updating semantic/graph-based representations can take advantage of recent advances in Bayesian reasoning as presented by *Pearl* [5.61], and exemplified in [5.62].

5.6 Conclusions and Further Readings

Sensing and estimation continues to be a challenging and very active area of robotics research. Several areas of sensing such as computer vision and medical imaging are themselves large and diverse research areas. At the same time, new fundamental and applied techniques in estimation continue to be developed. Indeed, it is fair to say that perception continues to be one of the most challenging areas of robotics research.

Given this wealth of activity, no single chapter can hope to cover all of the material that can be useful in the development of sensor-based robotics. However, the methods that have been presented here are representative of the most commonly used techniques in robotics. In particular, linear techniques such as the Kalman filter continue to form the backbone of perceptive robotics. Part C of the handbook provides more in-depth coverage of several of the key topics in sensing and estimation.

For the reader wishing to learn more, general discussion on the design, physics, and use of a rich variety of sensors can be found in the Handbook of Modern Sensors [5.3]. A discussion of sensors for mobile robots can be found in [5.63], though significant advances have been achieved since the book was published more than a decade ago. Sensing and estimation using computer vision is described in detail in [5.64] and [5.65].

The basic estimation theory is covered in a number of excellent text books. Much of the detection and linear estimation theory is covered in depth in [5.20] and [5.66]. General statistical estimation is covered in [5.12] and [5.13] and the more recently updated version [5.44]. Robust methods are described in detail in [5.21, 43]. In-depth coverage of estimation methods for mobile systems is also covered in [5.33].

References

5.1 D.C. Marr: *Vision* (Freeman, Bedford 1982)

5.2 R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza: *Introduction to Autonomous Mobile Robots*, Intelligent Robotics and Autonomous Systems (MIT Press, Cambridge 2011)

5.3 J. Fraden: *Handbook of Modern Sensors: Physic, Design and Applications*, 2nd edn. (Springer, New York 1996)

5.4 H. Yousef, M. Boukallel, K. Althoefer: Tactile sensing for dexterous in-hand manipulation in robotics – A review, Sensors Actuators A: Physical **167**(2), 171–187 (2011)

5.5 M.I. Tiwana, S.J. Redmond, N.H. Lovell: A review of tactile sensing technologies with applications in biomedical engineering, Sensors Actuators A: Physical **179**, 17–31 (2012)

5.6 G. Dissanayaka, S. Sukkarieh, E. Nebot, H. Durrant-Whyte: The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications, IEEE Trans. Robot. Autom. **17**(5), 731–748 (2001)

5.7 Z. Zhang: A flexible new technique for camera calibration, IEEE Trans. Pattern. Anal. Mach. Intell. **22**(11), 1330–1334 (2000)

5.8 D. Burschka, J. Geiman, G.D. Hager: Optimal landmark configuration for vision-based control of mobile robots, Proc. Int. Conf. Robot. Autom. ICRA (2003) pp. 3917–3922

5.9 J. Baker, N. Anderson, P. Pilles: Ground-penetrating radar surveying in support of archeological site investigations, Comput. Geosci. **23**(10), 1093–1099 (1997)

5.10 P.V.C. Hough: A method and means for recognizing complex patterns, U.S. Patent 306 9654 (1962)

5.11 M.A. Fischler, R.C. Bolles: Random Sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM **24**, 381–395 (1981)

5.12 T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning*, Springer Series in Statistics (Springer, Berlin, Heidelberg 2002)

5.13 R.O. Duda, P.E. Hart: *Pattern Classification and Scene Analysis* (Wiley-Interscience, New York 1973)

5.14 R. Vidal, Y. Ma, J. Piazzi: A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials, Proc. Int. Conf. Cumput. Vis. Pattern Recog. **1**, 510–517 (2004)

5.15 P. Besl, N.D. McKay: A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)

5.16 F. Dellaert, S. Seitz, C. Thorpe, S. Thrun: Special issue on Markov chain Monte Carlo methods, Mach. Learn. **50**, 45–71 (2003)

5.17 A. Gelb (Ed.): *Applied Optimal Estimation* (MIT Press, Cambridge 1974)

5.18 D. Simon: *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches* (Wiley, New York 2006)

5.19 A. Doucet, N. de Freitas, N. Gordon: *Sequential Monte Carlo Methods in Practice* (Springer, Berlin, Heidelberg 2001)

5.20 Y. Bar-Shalom, T. Fortmann: *Tracking and Data Association* (Academic, New York 1988)

5.21 P.J. Huber: *Robust Statistics* (Wiley, New York 1981)

5.22 J.O. Berger: *Statistical Decision Theory and Bayesian Analysis*, 2nd edn. (Springer, New York 1985)

5.23 G.D. Hager: *Task-Directed Sensor Fusion and Planning* (Kluwer, Boston 1990)

5.24 S. Abrams, P.K. Allen, K. Tarabanis: Computing camera viewpoints in a robot work-cell, Int. J. Robot. Res. **18**(3), 267–285 (1999)

5.25 M. Suppa, P. Wang, K. Gupta, G. Hirzinger: *C-space exploration using noisy sensor models, Proc* (IEEE, Int. Conf. Robot. Autom 2004) pp. 1927–1932

5.26 G.S. Chirikjian, A.B. Kyatkin: *Engineering Applications of Noncommutative Harmonic Analysis* (CRC, Boca Raton 2000)

5.27 J.C. Kinsey, L.L. Whitcomb: Adaptive identification on the group of rigid body rotations and its application to precision underwater robot navigation, IEEE Trans. Robot. **23**, 124–136 (2007)

5.28 P.J. Bickel, K.A. Doksum: *Mathematical Statistics*, 2nd edn. (Prentice-Hall, Upper Saddle River 2006)

5.29 G. Strang: *Linear Algebra and its Applications*, 4th edn. (Brooks Cole, New York 2005)

5.30 P. McCullagh, J.A. Nelder: *Generalized Linear Models*, 2nd edn. (Chapman Hall, New York 1989)

5.31 E.L. Lehmann, G. Casella: *Theory of Point Estimation* (Springer, New York 1998)

5.32 R.E. Kalman: A new approach to linear filtering and prediction problems, Transactions of the ASME, J. Basic Eng. **82**, 35–45 (1960)

5.33 S. Thrun, D. Fox, W. Burgard: *Probabilistic Robotics, Autonomous Robotics and Intelligent Agents* (MIT Press, Cambridge 2005)

5.34 C. Bishop: *Pattern Recognition and Machine Learning* (Springer, New York 2006)

5.35 T. Joachims, T. Finley, C.-N.J. Yu: Cutting-plane training of structural SVMs, Mach. Learn. **77**(1), 27–59 (2009)

5.36 S. Lacoste-Julien, M. Jaggi, M. Schmidt, P. Pletscher: Block-coordinate Frank-Wolfe optimization for structural SVMs, Proc. Int. Conf. Mach. Learn. (2013) pp. 53–61

5.37 L. Tao, L. Zappella, G.D. Hager, R. Vidal: Surgical gesture segmentation and recognition, Med. Image Comput. Comput.-Assisted Intervent., MICCAI 2013 (2013) pp. 339–346

5.38 C. Sutton, A. McCallum: An introduction to conditional random fields for relational learning. In: *Introduction to Statistical Relational Learning*, ed. by L. Getoor, B. Taskar (MIT Press, Cambridge 2006) pp. 93–128

5.39 C. Sutton, A. McCallum: An introduction to conditional random fields, Found. Trends Mach. Learn. **1**, 2055–2060 (2010)

5.40 C.A. Müller, S. Behnke: PyStruct-learning structured prediction in python, J. Mach. Learn. Res. **1**, 2055–2060 (2013)

5.41 J.W. Hardin, J.M. Hilbe: *Generalized Linear Models and Extensions*, 2nd edn. (Stata, College Station 2007)

5.42 G.D. Hager, P.N. Belhumeur: Efficient region tracking of with parametric models of illumination and geometry, IEEE Trans. Pattern Anal. Mach. Intell. **20**(10), 1025–1039 (1998)

5.43 P.J. Rousseauw, A. Leroy: *Robust Regression and Outlier Detection* (Wiley, New York 1987)

5.44 R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification*, 2nd edn. (Wiley, New York 2001)

5.45 P.C. Mahalanobis: On the generalised distance in statistics, Proc. Nat. Inst. Sci. India **12**, 49–55 (1936)

5.46 J. Hamilton: *Time Series Analysis* (Princeton Univ. Press, Princeton 1994)

5.47 S. Atiya, G.D. Hager: Real-time vision-based robot localization, IEEE Trans. Robot. Autom. **9**(6), 785–800 (1993)

5.48 G.D. Hager: Task-directed computation of qualitative decisions from sensor data, IEEE Trans. Robot. Autom. **10**(4), 415–429 (1994)

5.49 G. Shafer: *A Mathematical Theory of Evidence* (Princeton Univ. Press, Princeton 1976)

5.50 J. Rosenblatt: *DAMN: A distributed architecture for mobile navigation, AAAI 1995* (Spring, Symposium on Lessons Learned for Implementing Software Architectures for Physical Agents 1995) pp. 167–178

5.51 R.M. Murrey, Z. Li, S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1993)

5.52 K.J. Åström, B. Wittenmark: *Adaptive Control*, 2nd edn. (Addison-Wesley, Reading 1995)

5.53 S. Gutmann, C. Schlegel: AMOS: Comparison of scan-matching approaches for self-localization in indoor environments, 1st Euromicro Conf. Adv. Mobile Robotics (1996) pp. 61–67

5.54 S. Gutmann: *Robust Navigation for Autonomous Mobile Systems*, Ph.D. Thesis (Alfred Ludwig University, Freiburg 2000)

5.55 H. Samet: The quadtree and related hierarchical data structures, ACM Comput. Surv. **16**(2), 187–260 (1984)

5.56 A. Elfes: Sonar-based real-world mapping and navigation, IEEE Trans. Robot. Autom. **3**(3), 249–265 (1987)

5.57 A. Elfes: *A Probabilistic Framework for Robot Perception and Navigation*, Ph.D. Thesis (Carnegie Mellon University, Pittsburgh 1989)

5.58 M.R. Stytz, G. Frieder, O. Frieder: Three-dimensional medical imaging: Algorithms and computer systems, ACM Comput. Surv. **23**(4), 421–499 (1991)

5.59 R. Deriche, R. Vaillant, O. Faugeras: From Noisy Edges Points to 3D Reconstruction of a Scene: A robust approach and its uncertainty analysis. In: *Theory and Applications of Image Analysis*, (World Scientific, Singapore 1992) pp. 71–79

5.60 V.N. Vapnik: *Statistical Learning Theory* (Wiley, New York 1998)

5.61 J. Pearl: *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, New York 1988)

5.62 M. Paskin: *Thin Junction Tree Filters for Simultaneous Localisation and Mapping*, Ph.D. Thesis (University of California, Berkley 2002)

5.63 H.R. Everett: *Sensors for Mobile Robots: Theory and Application* (Peters, London 1995)

5.64 D. Forsyth, J. Ponce: *Computer Vision – A Modern Approach* (Prentice-Hall, Upper Saddle River 2003)

5.65 R. Hartley, A. Zisserman: *Multiple View Geometry in Computer Vision* (Cambridge Univ. Press, Cambridge 2000)

5.66 S. Blackman, R. Popoli: *Design and Analysis of Modern Tracking Systems* (Artech House, London 1999)