Roland SIEGWART
Illah R. NOURBAKHSH

Introduction to

Autonomous Mobile Robots

Introduction to Autonomous Mobile Robots

Roland Siegwart and Illah R. Nourbakhsh

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

5.5 Map Representation 200
5.5.1 Continuous representations 200
5.5.2 Decomposition strategies 203
5.5.3 State of the art: current challenges in map representation 210
5.6 Probabilistic Map-Based Localization 212
5.6.1 Introduction 212
5.6.2 Markov localization 214
5.6.3 Kalman filter localization 227
5.7 Other Examples of Localization Systems 244
5.7.1 Landmark-based navigation 245
5.7.2 Globally unique localization 246
5.7.3 Positioning beacon systems 248
5.7.4 Route-based localization 249
5.8 Autonomous Map Building 250
5.8.1 The stochastic map technique 250
5.8.2 Other mapping techniques 253

6 Planning and Navigation 257
6.1 Introduction 257
6.2 Competences for Navigation: Planning and Reacting 258
6.2.1 Path planning 259
6.2.2 Obstacle avoidance 272
6.3 Navigation Architectures 291
6.3.1 Modularity for code reuse and sharing 291
6.3.2 Control localization 291
6.3.3 Techniques for decomposition 292
6.3.4 Case studies: tiered robot architectures 298

Bibliography 305
Books 305
Papers 306
Referenced Webpages 314
Interesting Internet Links to Mobile Robots 314

Index 317

Figure 1.3
Plustech developed the first application-driven walking robot. It is designed to move wood out of the forest. The leg coordination is automated, but navigation is still done by the human operator on the robot. (http://www.plustech.fi). © Plustech.

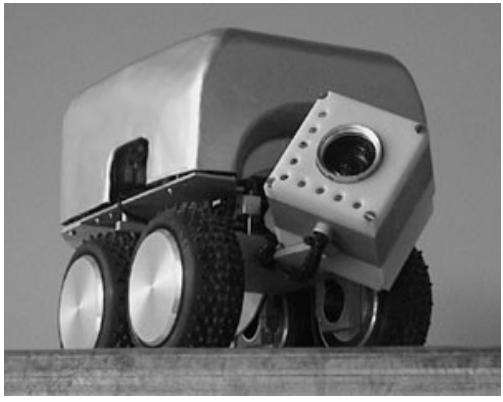


Figure 1.4
Airduct inspection robot featuring a pan-tilt camera with zoom and sensors for automatic inclination control, wall following, and intersection detection (http://asl.epfl.ch). © Sedirep / EPFL.

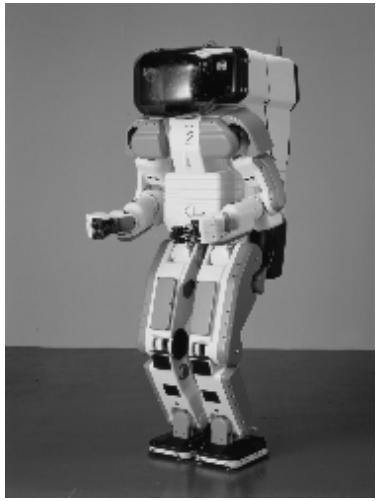
lar talents of each form of locomotion. But designing a robot's locomotive system properly requires the ability to evaluate its overall motion capabilities quantitatively. Chapter 3, "Mobile Robot Kinematics", applies principles of kinematics to the whole robot, beginning with the kinematic contribution of each wheel and graduating to an analysis of robot maneuverability enabled by each mobility mechanism configuration.

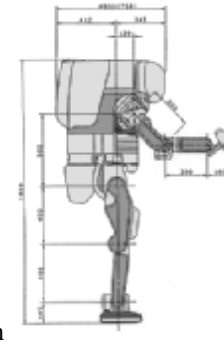
The greatest single shortcoming in conventional mobile robotics is, without doubt, perception: mobile robots can travel across much of earth's man-made surfaces, but they cannot perceive the world nearly as well as humans and other animals. Chapter 4, "Perception", begins a discussion of this challenge by presenting a clear language for describing the performance envelope of mobile robot sensors. With this language in hand, chapter 4 goes on to present many of the off-the-shelf sensors available to the mobile roboticist, describing their basic principles of operation as well as their performance limitations. The most promising sensor for the future of mobile robotics is vision, and chapter 4 includes an overview of the theory of operation and the limitations of both charged coupled device (CCD) and complementary metal oxide semiconductor (CMOS) sensors.

But perception is more than sensing. Perception is also the *interpretation* of sensed data in meaningful ways. The second half of chapter 4 describes strategies for feature extraction that have been most useful in mobile robotics applications, including extraction of geometric shapes from range-based sensing data, as well as landmark and whole-image analysis using vision-based sensing.

Armed with locomotion mechanisms and outfitted with hardware and software for perception, the mobile robot can move and perceive the world. The first point at which mobility and sensing must meet is localization: mobile robots often need to maintain a sense of position. Chapter 5, "Mobile Robot Localization", describes approaches that obviate the need for direct localization, then delves into fundamental ingredients of successful localization strategies: belief representation and map representation. Case studies demonstrate various localization schemes, including both Markov localization and Kalman filter localization. The final part of chapter 5 is devoted to a discussion of the challenges and most promising techniques for mobile robots to autonomously map their surroundings.

Mobile robotics is so young a discipline that it lacks a standardized architecture. There is as yet no established robot operating system. But the question of architecture is of paramount importance when one chooses to address the higher-level competences of a mobile robot: how does a mobile robot navigate robustly from place to place, interpreting data, localizing and controlling its motion all the while? For this highest level of robot competence, which we term *navigation competence*, there are numerous mobile robots that showcase particular architectural strategies. Chapter 6, "Planning and Navigation", surveys the state of the art of robot navigation, showing that today's various techniques are quite similar, differing primarily in the manner in which they *decompose* the problem of robot con-





Specifications:

Maximum speed:	2 km/h
Autonomy:	15 min
Weight:	210 kg
Height:	1.82 m
Leg DOF:	2 x 6
Arm DOF:	2 x 7

Figure 2.12
The humanoid robot P2 from Honda, Japan. © Honda Motor Corporation.

The Sony Dream Robot, model SDR-4X II, is shown in figure 2.11. This current model is the result of research begun in 1997 with the basic objective of motion entertainment and communication entertainment (i.e., dancing and singing). This robot with thirty-eight degrees of freedom has seven microphones for fine localization of sound, image-based person recognition, on-board miniature stereo depth-map reconstruction, and limited speech recognition. Given the goal of fluid and entertaining motion, Sony spent considerable effort designing a motion prototyping application system to enable their engineers to script dances in a straightforward manner. Note that the SDR-4X II is relatively small, standing at 58 cm and weighing only 6.5 kg.

The Honda humanoid project has a significant history but, again, has tackled the very important engineering challenge of actuation. Figure 2.12 shows model *P2*, which is an immediate predecessor to the most recent Asimo model (advanced step in innovative mobility). Note from this picture that the Honda humanoid is much larger than the SDR-4X at 120 cm tall and 52 kg. This enables practical mobility in the human world of stairs and ledges while maintaining a nonthreatening size and posture. Perhaps the first robot to famously demonstrate biomimetic bipedal stair climbing and descending, these Honda humanoid series robots are being designed not for entertainment purposes but as human aids throughout society. Honda refers, for instance, to the height of Asimo as the minimum height which enables it to nonetheless manage operation of the human world, for instance, control of light switches.

Table 2.1
Wheel configurations for rolling vehicles

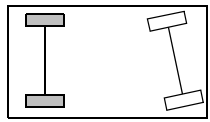
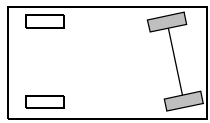
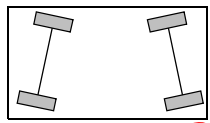
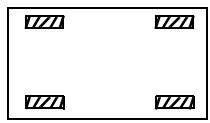
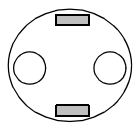
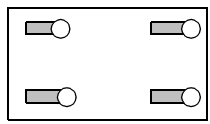
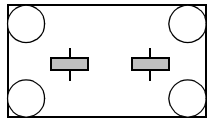
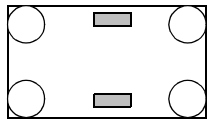
# of wheels	Arrangement	Description	Typical examples
4		Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with rear-wheel drive
		Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with front-wheel drive
		Four steered and motorized wheels	four-wheel drive, four-wheel steering Hyperion (CMU)
		Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear	Charlie (DMT-EPFL)
		Four omnidirectional wheels	Carnegie Mellon Uranus
		Two-wheel differential drive with 2 additional points of contact	EPFL Khepera, Hyperbot Chip
		Four motorized and steered castor wheels	Nomad XR4000

Table 2.1
Wheel configurations for rolling vehicles

# of wheels	Arrangement	Description	Typical examples
6		Two motorized and steered wheels aligned in center, 1 omnidirectional wheel at each corner	First
		Two traction wheels (differential) in center, 1 omnidirectional wheel at each corner	Terregator (Carnegie Mellon University)
Icons for the each wheel type are as follows:			
	unpowered omnidirectional wheel (spherical, castor, Swedish);		
	motorized Swedish wheel (Stanford wheel);		
	unpowered standard wheel;		
	motorized standard wheel;		
	motorized and steered castor wheel;		
	steered standard wheel;		
	connected wheels.		

2.3.1.4 Maneuverability

Some robots are omnidirectional, meaning that they can move at any time in any direction along the ground plane (x, y) regardless of the orientation of the robot around its vertical axis. This level of maneuverability requires wheels that can move in more than just one direction, and so omnidirectional robots usually employ Swedish or spherical wheels that are powered. A good example is Uranus, shown in figure 2.24. This robot uses four Swedish wheels to rotate and translate independently and without constraints.

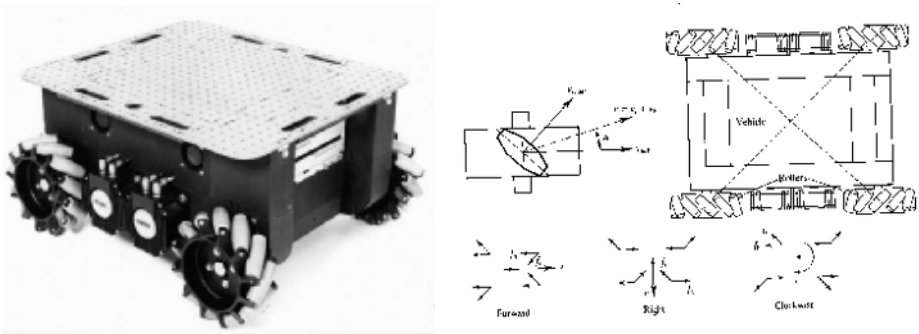


Figure 2.24
The Carnegie Mellon Uranus robot, an omnidirectional robot with four powered-swedish 45 wheels.

For example, when all four wheels spin "forward" or "backward" the robot as a whole moves in a straight line forward or backward, respectively. However, when one diagonal pair of wheels is spun in the same direction and the other diagonal pair is spun in the opposite direction, the robot moves laterally.

This four-wheel arrangement of Swedish wheels is not minimal in terms of control motors. Because there are only three degrees of freedom in the plane, one can build a three-wheel omnidirectional robot chassis using three Swedish 90-degree wheels as shown in table 2.1. However, existing examples such as Uranus have been designed with four wheels owing to capacity and stability considerations.

One application for which such omnidirectional designs are particularly amenable is mobile manipulation. In this case, it is desirable to reduce the degrees of freedom of the manipulator arm to save arm mass by using the mobile robot chassis motion for gross motion. As with humans, it would be ideal if the base could move omnidirectionally without greatly impacting the position of the manipulator tip, and a base such as Uranus can afford precisely such capabilities.

Omnidirectional locomotion with four castor wheels and eight motors. Another solution for omnidirectionality is to use castor wheels. This is done for the Nomad XR4000 from Nomadic Technologies (fig. 2.25), giving it excellent maneuverability. Unfortunately, Nomadic has ceased production of mobile robots.

The above three examples are drawn from table 2.1, but this is not an exhaustive list of all wheeled locomotion techniques. Hybrid approaches that combine legged and wheeled locomotion, or tracked and wheeled locomotion, can also offer particular advantages. Below are two unique designs created for specialized applications.

$$\varphi(t) = \begin{bmatrix} \varphi_f(t) \\ \varphi_s(t) \end{bmatrix} \tag{3.23}$$

The rolling constraints of all wheels can now be collected in a single expression:

$$J_1(\beta_s)R(\theta)\dot{\xi}_I - J_2\dot{\varphi} = 0 \tag{3.24}$$

This expression bears a strong resemblance to the rolling constraint of a single wheel, but substitutes matrices in lieu of single values, thus taking into account all wheels. J_2 is a constant diagonal $N \times N$ matrix whose entries are radii r of all standard wheels. $J_1(\beta_s)$ denotes a matrix with projections for all wheels to their motions along their individual wheel planes:

$$J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix} \tag{3.25}$$

Note that $J_{1s}(\beta_s)$ is only a function of β_s and not β_f. This is because the orientations of steerable standard wheels vary as a function of time, whereas the orientations of fixed standard wheels are constant. J_{1f} is therefore a constant matrix of projections for all fixed standard wheels. It has size $(N_f \times 3)$, with each row consisting of the three terms in the three-matrix from equation (3.12) for each fixed standard wheel. $J_{1s}(\beta_s)$ is a matrix of size $(N_s \times 3)$, with each row consisting of the three terms in the three-matrix from equation (3.15) for each steerable standard wheel.

In summary, equation (3.24) represents the constraint that all standard wheels must spin around their horizontal axis an appropriate amount based on their motions along the wheel plane so that rolling occurs at the ground contact point.

We use the same technique to collect the sliding constraints of all standard wheels into a single expression with the same structure as equations (3.13) and (3.16):

$$C_1(\beta_s)R(\theta)\dot{\xi}_I = 0 \tag{3.26}$$

$$C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix} \tag{3.27}$$

C_{1f} and C_{1s} are $(N_f \times 3)$ and $(N_s \times 3)$ matrices whose rows are the three terms in the three-matrix of equations (3.13) and (3.16) for all fixed and steerable standard wheels. Thus

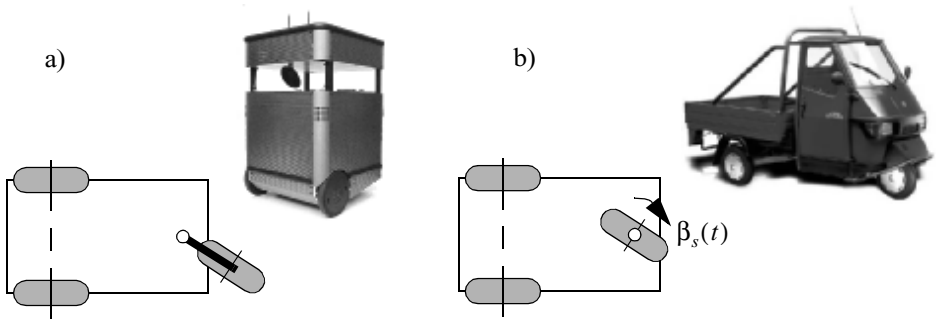


Figure 3.13
(a) Differential drive robot with two individually motorized wheels and a castor wheel, e.g., the Pygmalion robot at EPFL. (b) Tricycle with two fixed standard wheels and one steered standard wheel, e.g. Piaggio minitransporter.

The Ackerman vehicle of figure 3.12a demonstrates another way in which a wheel may be unable to contribute an independent constraint to the robot kinematics. This vehicle has two steerable standard wheels. Given the instantaneous position of just one of these steerable wheels and the position of the fixed rear wheels, there is only a single solution for the *ICR*. The position of the second steerable wheel is absolutely constrained by the *ICR*. Therefore, it offers no independent constraints to robot motion.

Robot chassis kinematics is therefore a function of the set of *independent* constraints arising from all standard wheels. The mathematical interpretation of independence is related to the *rank* of a matrix. Recall that the rank of a matrix is the smallest number of independent rows or columns. Equation (3.26) represents all sliding constraints imposed by the wheels of the mobile robot. Therefore $rank\left[C_1(\beta_s)\right]$ is the number of independent constraints.

The greater the number of independent constraints, and therefore the greater the rank of $C_1(\beta_s)$, the more constrained is the mobility of the robot. For example, consider a robot with a single fixed standard wheel. Remember that we consider only standard wheels. This robot may be a unicycle or it may have several Swedish wheels; however, it has exactly one fixed standard wheel. The wheel is at a position specified by parameters α, β, l relative to the robot's local reference frame. $C_1(\beta_s)$ is comprised of C_{1f} and C_{1s}. However, since there are no steerable standard wheels C_{1s} is empty and therefore $C_1(\beta_s)$ contains only C_{1f}. Because there is one fixed standard wheel, this matrix has a rank of one and therefore this robot has a single independent constrain on mobility:

$$C_1(\beta_s) = C_{1f} = \left[\cos(\alpha+\beta) \;\; \sin(\alpha+\beta) \;\; l\sin\beta\right] \tag{3.37}$$

Now let us add an additional fixed standard wheel to create a differential-drive robot by constraining the second wheel to be aligned with the same horizontal axis as the original wheel. Without loss of generality, we can place point P at the midpoint between the centers of the two wheels. Given α_1, β_1, l_1 for wheel w_1 and α_2, β_2, l_2 for wheel w_2, it holds geometrically that $\{(l_1 = l_2), (\beta_1 = \beta_2 = 0), (\alpha_1 + \pi = \alpha_2)\}$. Therefore, in this case, the matrix $C_1(\beta_s)$ has two constraints but a rank of one:

$$C_1(\beta_s) = C_{1f} = \begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) & 0 \\ \cos(\alpha_1 + \pi) & \sin(\alpha_1 + \pi) & 0 \end{bmatrix} \tag{3.38}$$

Alternatively, consider the case when w_2 is placed in the wheel plane of w_1 but with the same orientation, as in a bicycle with the steering locked in the forward position. We again place point P between the two wheel centers, and orient the wheels such that they lie on axis x_1. This geometry implies that $\{(l_1 = l_2), (\beta_1 = \beta_2 = \pi/2), (\alpha_1 = 0), (\alpha_2 = \pi)\}$ and, therefore, the matrix $C_1(\beta_s)$ retains two independent constraints and has a rank of two:

$$C_1(\beta_s) = C_{1f} = \begin{bmatrix} \cos(\pi/2) & \sin(\pi/2) & l_1 \sin(\pi/2) \\ \cos(3\pi/2) & \sin(3\pi/2) & l_1 \sin(\pi/2) \end{bmatrix} = \begin{bmatrix} 0 & 1 & l_1 \\ 0 & -1 & l_1 \end{bmatrix} \tag{3.39}$$

In general, if $rank\left[C_{1f}\right] > 2$ then the vehicle can, at best, only travel along a circle or along a straight line. This configuration means that the robot has two or more independent constraints due to fixed standard wheels that do not share the same horizontal axis of rotation. Because such configurations have only a degenerate form of mobility in the plane, we do not consider them in the remainder of this chapter. Note, however, that some degenerate configurations such as the four-wheeled slip/skid steering system are useful in certain environments, such as on loose soil and sand, even though they fail to satisfy sliding constraints. Not surprisingly, the price that must be paid for such violations of the sliding constraints is that dead reckoning based on odometry becomes less accurate and power efficiency is reduced dramatically.

In general, a robot will have zero or more fixed standard wheels and zero or more steerable standard wheels. We can therefore identify the possible range of rank values for any robot: $0 \le rank\left[C_1(\beta_s)\right] \le 3$. Consider the case $rank\left[C_1(\beta_s)\right] = 0$. This is only possible if there are zero independent kinematic constraints in $C_1(\beta_s)$. In this case there are neither fixed nor steerable standard wheels attached to the robot frame: $N_f = N_s = 0$.

Consider the other extreme, $rank\left[C_1(\beta_s)\right] = 3$. This is the maximum possible rank since the kinematic constraints are specified along three degrees of freedom (i.e., the constraint matrix is three columns wide). Therefore, there cannot be more than three indepen-

An alternative way to describe a holonomic robot is based on the relationship between the differential degrees of freedom of a robot and the degrees of freedom of its workspace: *a robot is holonomic if and only if* $DDOF = DOF$. Intuitively, this is because it is only through nonholonomic constraints (imposed by steerable or fixed standard wheels) that a robot can achieve a workspace with degrees of freedom exceeding its differential degrees of freedom, $DOF > DDOF$. Examples include differential drive and bicycle/tricycle configurations.

In mobile robotics, useful chassis generally must achieve poses in a workspace with dimensionality 3, so in general we require $DOF = 3$ for the chassis. But the "holonomic" abilities to maneuver around obstacles without affecting orientation and to track at a target while following an arbitrary path are important additional considerations. For these reasons, the particular form of holonomy most relevant to mobile robotics is that of $DDOF = DOF = 3$. We define this class of robot configurations as omnidirectional: an *omnidirectional robot* is a holonomic robot with $DDOF = 3$.

3.4.3 Path and trajectory considerations

In mobile robotics, we care not only about the robot's ability to reach the required final configurations but also about *how* it gets there. Consider the issue of a robot's ability to follow paths: in the best case, a robot should be able to trace any path through its workspace of poses. Clearly, any omnidirectional robot can do this because it is holonomic in a three-dimensional workspace. Unfortunately, omnidirectional robots must use unconstrained wheels, limiting the choice of wheels to Swedish wheels, castor wheels, and spherical wheels. These wheels have not yet been incorporated into designs allowing far larger amounts of ground clearance and suspensions. Although powerful from a path space point of view, they are thus much less common than fixed and steerable standard wheels, mainly because their design and fabrication are somewhat complex and expensive.

Additionally, nonholonomic constraints might drastically improve stability of movements. Consider an omnidirectional vehicle driving at high speed on a curve with constant diameter. During such a movement the vehicle will be exposed to a non-negligible centripetal force. This lateral force pushing the vehicle out of the curve has to be counteracted by the motor torque of the omnidirectional wheels. In case of motor or control failure, the vehicle will be thrown out of the curve. However, for a car-like robot with kinematic constraints, the lateral forces are passively counteracted through the sliding constraints, mitigating the demands on motor torque.

But recall an earlier example of high maneuverability using standard wheels: the bicycle on which both wheels are steerable, often called the *two-steer*. This vehicle achieves a degree of steerability of 2, resulting in a high degree of maneuverability: $\delta_M = \delta_m + \delta_s = 1 + 2 = 3$. Interestingly, this configuration is not holonomic, yet has a high degree of maneuverability in a workspace with $DOF = 3$.

θ denotes the angle between the X_R axis of the robot reference frame, and the X_I axis associated with the final position v and ω are the tangent and the angular velocity respectively.

On the other hand, if $\alpha \in I_2$, where

$$I_2 = (-\pi, -\pi/2] \cup (\pi/2, \pi] \tag{3.54}$$

redefining the forward direction of the robot by setting $v = -v$, we obtain a system described by a matrix equation of the form

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 \\ -\frac{\sin\alpha}{\rho} & 1 \\ \frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.55}$$

3.6.2.3 Remarks on the kinematic model in polar coordinates [eq. (3.53) and (3.55)]

- The coordinate transformation is not defined at $x = y = 0$; as in such a point the determinant of the Jacobian matrix of the transformation is not defined, that is unbounded.
- For $\alpha \in I_1$ the forward direction of the robot points toward the goal, for $\alpha \in I_2$ it is the reverse direction.
- By properly defining the forward direction of the robot at its initial configuration, it is always possible to have $\alpha \in I_1$ at $t = 0$. However, this does not mean that α remains in I_I for all time t. Hence, to avoid that the robot changes direction during approaching the goal, it is necessary to determine, if possible, the controller in such a way that $\alpha \in I_1$ for all t, whenever $\alpha(0) \in I_1$. The same applies for the reverse direction (see stability issues below).

3.6.2.4 The control law

The control signals v and ω must now be designed to drive the robot from its actual configuration, say $(\rho_0, \alpha_0, \beta_0)$, to the goal position. It is obvious that equation (3.53) presents a discontinuity at $\rho = 0$; thus the theorem of Brockett does not obstruct smooth stabilizability.

If we consider now the linear control law

$$v = k_\rho \rho \tag{3.56}$$

$$\omega = k_\alpha \alpha + k_\beta \beta \tag{3.57}$$

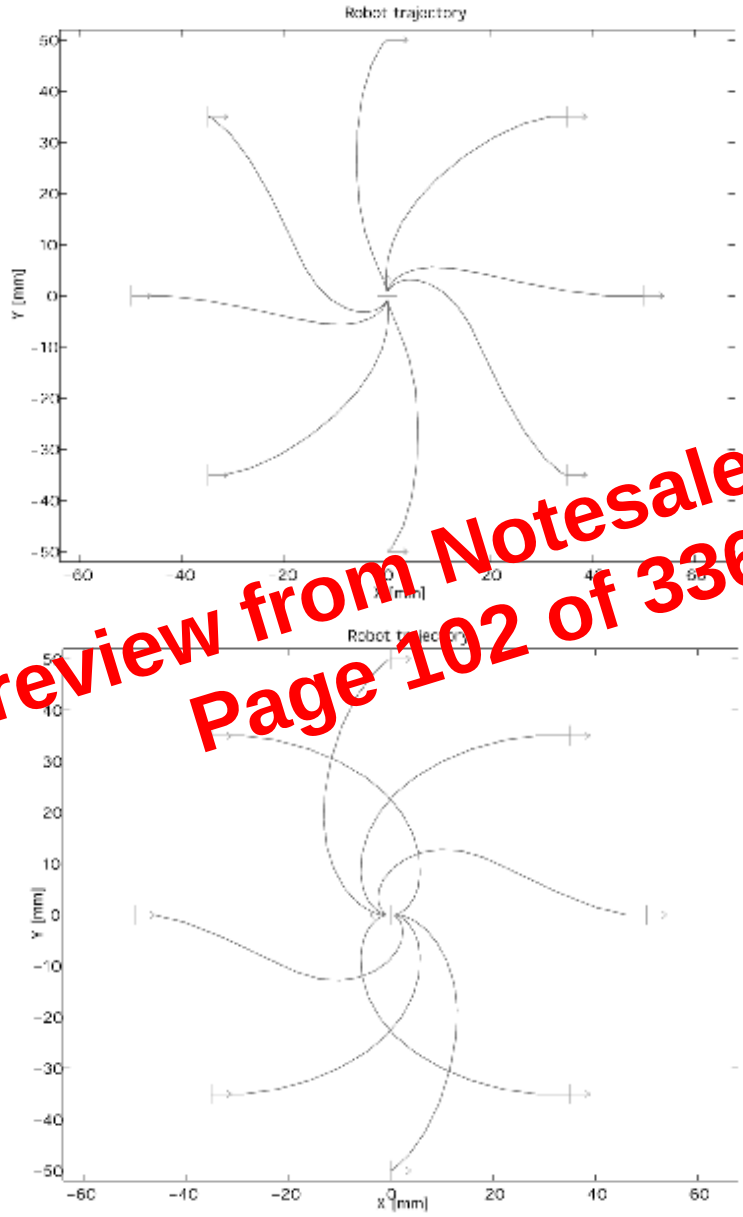


Figure 3.20
Resulting paths when the robot is initially on the unit circle in the *x,y* plane.

4 Perception

One of the most important tasks of an autonomous system of any kind is to acquire knowledge about its environment. This is done by taking measurements using various sensors and then extracting meaningful information from those measurements.

In this chapter we present the most common sensors used in mobile robots and then discuss strategies for extracting information from the sensors. For more detailed information about many of the sensors used on mobile robots, we refer to the comprehensive book *Sensors for Mobile Robots* by H.R. Everett [11].

4.1 Sensors for Mobile Robots

There are a wide variety of sensors used in mobile robots (figure 4.1). Some sensors are used to measure simple values like the internal temperature of a robot's electronics or the rotational speed of the motors. Other, more sophisticated sensors can be used to acquire information about the robot's environment or even to directly measure a robot's global position. In this chapter we focus primarily on sensors used to extract information about the robot's environment. Because a mobile robot moves around, it will frequently encounter unforeseen environmental characteristics, and therefore such sensing is particularly critical. We begin with a functional classification of sensors. Then, after presenting basic tools for describing a sensor's performance, we proceed to describe selected sensors in detail.

4.1.1 Sensor classification

We classify sensors using two important functional axes: *proprioceptive/exteroceptive* and *passive/active*.

Proprioceptive sensors measure values internal to the system (robot); for example, motor speed, wheel load, robot arm joint angles, battery voltage.

Exteroceptive sensors acquire information from the robot's environment; for example, distance measurements, light intensity, sound amplitude. Hence exteroceptive sensor measurements are interpreted by the robot in order to extract meaningful environmental features.

Rate gyros have the same basic arrangement as shown in figure 4.4 but with a slight modification. The gimbals are restrained by a torsional spring with additional viscous damping. This enables the sensor to measure angular speeds instead of absolute orientation.

Optical gyroscopes. Optical gyroscopes are a relatively new innovation. Commercial use began in the early 1980s when they were first installed in aircraft. Optical gyroscopes are angular speed sensors that use two monochromatic light beams, or lasers, emitted from the same source, instead of moving, mechanical parts. They work on the principle that the speed of light remains unchanged and, therefore, geometric change can cause light to take a varying amount of time to reach its destination. One laser beam is sent traveling clockwise through a fiber while the other travels counterclockwise. Because the laser traveling in the direction of rotation has a slightly shorter path, it will have a higher frequency. The difference in frequency Δf of the two beams is a proportional to the angular velocity Ω of the cylinder. New solid-state optical gyroscopes based on the same principle are being built using microfabrication technology, thereby providing heading information with resolution and bandwidth far beyond the needs of mobile robotic applications. Bandwidth, for instance, can easily exceed 100 kHz while resolution can be smaller than 0.0001 degrees/hr.

4.1.5 Ground-based beacons

One elegant approach to solving the localization problem in mobile robotics is to use active or passive beacons. Using the interaction of on-board sensors and the environmental beacons, the robot can identify its position precisely. Although the general intuition is identical to that of early human navigation beacons, such as stars, mountains, and lighthouses, modern technology has enabled sensors to localize an outdoor robot with accuracies of better than 5 cm within areas that are kilometers in size.

In the following section, we describe one such beacon system, the global positioning system (GPS), which is extremely effective for outdoor ground-based and flying robots. Indoor beacon systems have been generally less successful for a number of reasons. The expense of environmental modification in an indoor setting is not amortized over an extremely large useful area, as it is, for example, in the case of the GPS. Furthermore, indoor environments offer significant challenges not seen outdoors, including multipath and environmental dynamics. A laser-based indoor beacon system, for example, must disambiguate the one true laser signal from possibly tens of other powerful signals that have reflected off of walls, smooth floors, and doors. Confounding this, humans and other obstacles may be constantly changing the environment, for example, occluding the one true path from the beacon to the robot. In commercial applications, such as manufacturing plants, the environment can be carefully controlled to ensure success. In less structured indoor settings, beacons have nonetheless been used, and the problems are mitigated by careful beacon placement and the use of passive sensing modalities.

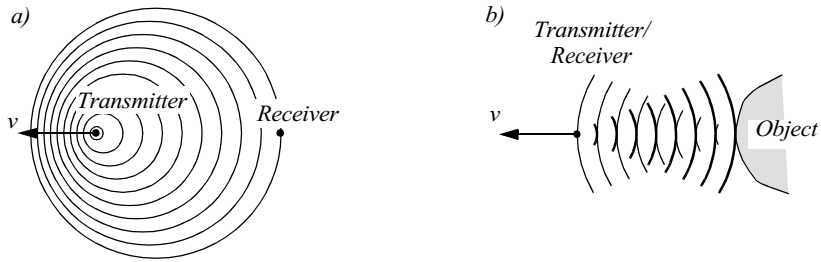


Figure 4.16
Doppler effect between two moving objects (a) or a moving and a stationary object (b).

of view, his or her motion triggers a change in heat in the sensor's reference frame. In the next section, we describe an important type of motion detector based on the Doppler effect. These sensors represent a well-known technology with decades of general applications behind them. For fast-moving mobile robots such as autonomous highway vehicles and unmanned flying vehicles, Doppler-based motion detectors are the obstacle detection sensor of choice.

4.1.7.1 Doppler effect-based sensing (radar or sound)

Anyone who has noticed the change in siren pitch that occurs when an approaching fire engine passes by and recedes is familiar with the Doppler effect.

A transmitter emits an electromagnetic or sound wave with a frequency f_t. It is either received by a receiver (figure 4.16a) or reflected from an object (figure 4.16b). The measured frequency f_r at the receiver is a function of the relative speed v between transmitter and receiver according to

$$f_r = f_t \frac{1}{1 + v/c} \tag{4.15}$$

if the transmitter is moving and

$$f_r = f_t (1 + v/c) \tag{4.16}$$

if the receiver is moving.

In the case of a reflected wave (figure 4.16b) there is a factor of 2 introduced, since any change x in relative separation affects the round-trip path length by $2x$. Furthermore, in such situations it is generally more convenient to consider the change in frequency Δf, known as the *Doppler shift*, as opposed to the *Doppler frequency* notation above.

Camera output considerations. Although digital cameras have inherently digital output, throughout the 1980s and early 1990s, most affordable vision modules provided analog output signals, such as NTSC (National Television Standards Committee) and PAL (Phase Alternating Line). These camera systems included a D/A converter which, ironically, would be counteracted on the computer using a *framegrabber*, effectively an A/D converter board situated, for example, on a computer's bus. The D/A and A/D steps are far from noisefree, and furthermore the color depth of the analog signal in such cameras was optimized for human vision, not computer vision.

More recently, both CCD and CMOS technology vision systems provide digital signals that can be directly utilized by the roboticist. At the most basic level, an imaging chip provides parallel digital I/O (input/output) pins that communicate discrete pixel level values. Some vision modules make use of these direct digital signals, which must be handled subject to hard-time constraints governed by the imaging chip. To relieve the real-time demands, researchers often place an *image buffer chip* between the imager's digital output and the computer's digital inputs. Such chips, commonly used in webcams, capture a complete image snapshot and enable non real time access to the pixels, usually in a single, ordered pass.

At the highest level, a roboticist may choose instead to utilize a higher-level digital transport protocol to communicate with an imager. Most common are the IEEE 1394 (*Firewire*, i.Link) and the USB (and USB 2.0) standards, although some older imaging modules also support serial (RS-232). To use any such high-level protocol, one must locate or create driver code both for that communication layer and for the particular implementation details of the imaging chip. Take note, however, of the distinction between lossless digital video and the standard digital video stream designed for human visual consumption. Most digital video cameras provide digital output, but often only in compressed form. For vision researchers, such compression must be avoided as it not only discards information but even introduces image detail that does not actually exist, such as MPEG (Moving Picture Experts Group) discretization boundaries.

4.1.8.2 Visual ranging sensors

Range sensing is extremely important in mobile robotics as it is a basic input for successful obstacle avoidance. As we have seen earlier in this chapter, a number of sensors are popular in robotics explicitly for their ability to recover depth estimates: ultrasonic, laser rangefinder, optical rangefinder, and so on. It is natural to attempt to implement ranging functionality using vision chips as well.

However, a fundamental problem with visual images makes rangefinding relatively difficult. Any vision chip collapses the 3D world into a 2D image plane, thereby losing depth information. If one can make strong assumptions regarding the size of objects in the world, or their particular color and reflectance, then one can directly interpret the appearance of the 2D image to recover depth. But such assumptions are rarely possible in real-world

In order to carry out the calibration step of step 2 above, we must find values for twelve unknowns, requiring twelve equations. This means that calibration requires, for a given scene, four conjugate points.

The above example supposes that regular translation and rotation are all that are required to effect sufficient calibration for stereo depth recovery using two cameras. In fact, single-camera calibration is itself an active area of research, particularly when the goal includes any 3D recovery aspect. When researchers intend to use even a single camera with high precision in 3D, internal errors relating to the exact placement of the imaging chip relative to the lens optical axis, as well as aberrations in the lens system itself, must be calibrated against. Such single-camera calibration involves finding solutions for the values for the exact offset of the imaging chip relative to the optical axis, both in translation and angle, and finding the relationship between distance along the imaging chip surface and external viewed surfaces. Furthermore, even without optical aberration in play, the lens is an inherently radial instrument, and so the image projected upon a flat imaging surface is radially distorted (i.e., parallel lines in the viewed world converge on the imaging chip).

A commonly practiced technique for such single-camera calibration is based upon acquiring multiple views of an easily analyzed planar pattern, such as a grid of black squares on a white background. The corners of such squares can easily be extracted, and using an interactive refinement algorithm the intrinsic calibration parameters of a camera can be extracted. Because modern imaging systems are capable of spatial accuracy greatly exceeding the pixel size, the payoff of such refined calibration can be significant. For further discussion of calibration and to download and use a standard calibration program, see [158].

Assuming that the calibration step is complete, we can now formalize the range recovery problem. To begin with, we do not have the position of P available, and therefore (x'_l, y'_l, z'_l) and (x'_r, y'_r, z'_r) are unknowns. Instead, by virtue of the two cameras we have pixels on the image planes of each camera, (x_l, y_l, z_l) and (x_r, y_r, z_r). Given the focal length f of the cameras we can relate the position of P to the left camera image as follows:

$$\frac{x_l}{f} = \frac{x'_l}{z'_l} \text{ and } \frac{y_l}{f} = \frac{y'_l}{z'_l} \tag{4.31}$$

Let us concentrate first on recovery of the values z'_l and z'_r. From equations (4.30) and (4.31) we can compute these values from any two of the following equations:

$$\left(r_{11}\frac{x_l}{f} + r_{12}\frac{y_l}{f} + r_{13}\right)z'_l + r_{01} = \frac{x_r}{f}z'_r \tag{4.32}$$

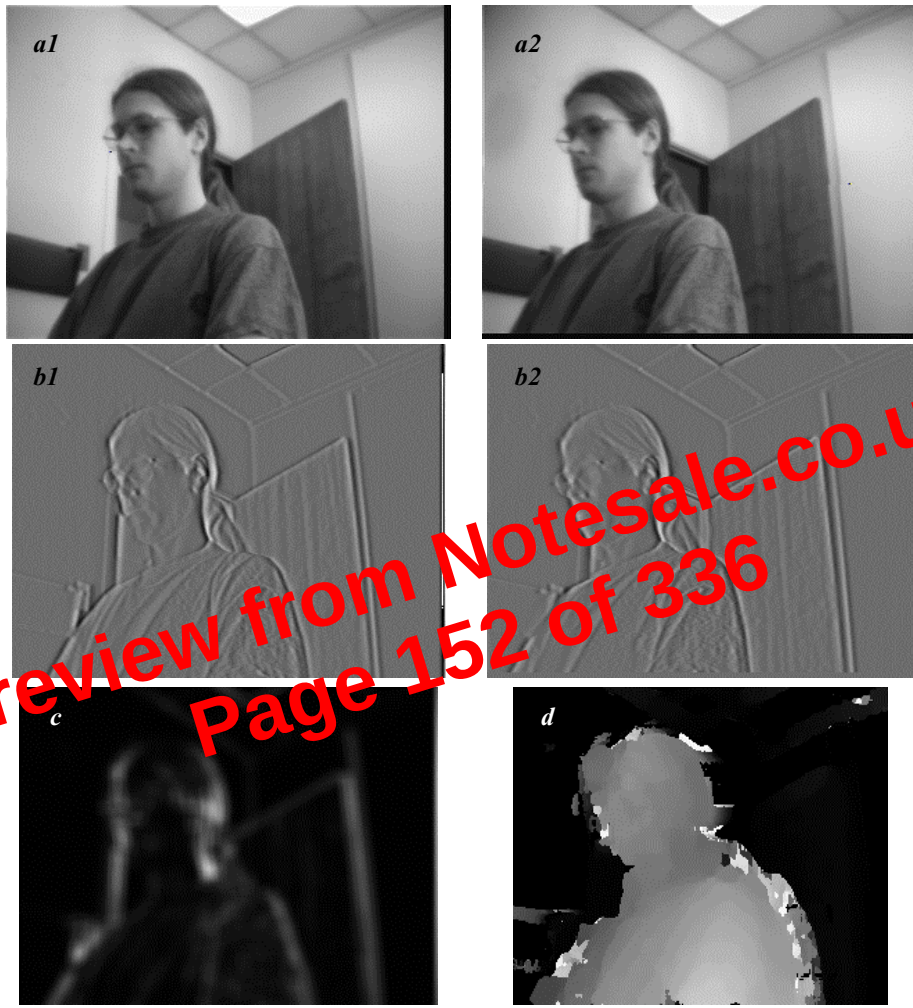


Figure 4.24
Extracting depth information from a stereo image. (a1 and a2) Left and right image. (b1 and b2) Vertical edge filtered left and right image: filter = [1 2 4 -2 -10 -2 4 2 1]. (c) Confidence image: bright = high confidence (good texture); dark = low confidence (no texture). (d) Depth image (disparity): bright = close; dark = far.

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \tag{4.41}$$

from which we can abbreviate

$$u = \frac{dx}{dt} \; ; \quad v = \frac{dy}{dt} \tag{4.42}$$

and

$$E_x = \frac{\partial E}{\partial x} \; ; \quad E_y = \frac{\partial E}{\partial y} ; \quad E_t = \frac{\partial E}{\partial t} = 0 \tag{4.43}$$

so that we obtain

$$E_x u + E_y v + E_t = 0 \tag{4.44}$$

The derivative E_t represents how quickly the intensity changes with time while the derivatives E_x and E_y represent the spatial rates of intensity change (how quickly intensity changes across the image). Altogether, equation (4.44) is known as the *optical flow constraint equation* and the three derivatives can be estimated for each pixel given successive images.

We need to calculate both u and v for each pixel, but the optical flow constraint equation only provides one equation per pixel, and so this is insufficient. The ambiguity is intuitively clear when one considers that a number of equal-intensity pixels can be inherently ambiguous – it may be unclear which pixel is the resulting location for an equal-intensity originating pixel in the prior image.

The solution to this ambiguity requires an additional constraint. We assume that in general the motion of adjacent pixels will be similar, and that therefore the overall optical flow of all pixels will be smooth. This constraint is interesting in that we know it will be violated to *some* degree, but we enforce the constraint nonetheless in order to make the optical flow computation tractable. Specifically, this constraint will be violated precisely when different objects in the scene are moving in different directions with respect to the vision system. Of course, such situations will tend to include edges, and so this may introduce a useful visual cue.

Because we know that this smoothness constraint will be somewhat incorrect, we can mathematically define the degree to which we violate this constraint by evaluating the formula

In the following two sections, we present specific feature extraction techniques based on the two most popular sensing modalities of mobile robotics: range sensing and visual appearance-based sensing.

4.3.1 Feature extraction based on range data (laser, ultrasonic, vision-based ranging)

Most of today's features extracted from ranging sensors are geometric primitives such as line segments or circles. The main reason for this is that for most other geometric primitives the parametric description of the features becomes too complex and no closed-form solution exists. Here we describe line extraction in detail, demonstrating how the uncertainty models presented above can be applied to the problem of combining multiple sensor measurements. Afterward, we briefly present another very successful feature of indoor mobile robots, the corner feature, and demonstrate how these features can be combined in a single representation.

4.3.1.1 Line extraction

Geometric feature extraction is usually the process of comparing and matching measured sensor data against a predefined description or template, of the expected feature. Usually, the system is overdetermined in that the number of sensor measurements exceeds the number of feature parameters to be estimated. Since the sensor measurements all have some error, there is no perfectly consistent solution and, instead, the problem is one of optimization. One can, for example, extract the feature that minimizes the discrepancy with all sensor measurements used (e.g., least squares estimation).

In this section we present an optimization-based solution to the problem of extracting a line feature from a set of uncertain sensor measurements. For greater detail than is presented below, refer to [14, pp. 15 and 221].

Probabilistic line extraction from uncertain range sensor data. Our goal is to extract a line feature based on a set of sensor measurements as shown in figure 4.36. There is uncertainty associated with each of the noisy range sensor measurements, and so there is no single line that passes through the set. Instead, we wish to select the best possible match, given some optimization criterion.

More formally, suppose n ranging measurement points in polar coordinates $x_i = (\rho_i, \theta_i)$ are produced by the robot's sensors. We know that there is uncertainty associated with each measurement, and so we can model each measurement using two random variables $X_i = (P_i, Q_i)$. In this analysis we assume that uncertainty with respect to the actual value of P and Q is independent. Based on equation (4.56) we can state this formally:

$$E[P_i \cdot P_j] = E[P_i]E[P_j] \qquad \forall\, i, j = 1, \ldots, n \tag{4.62}$$

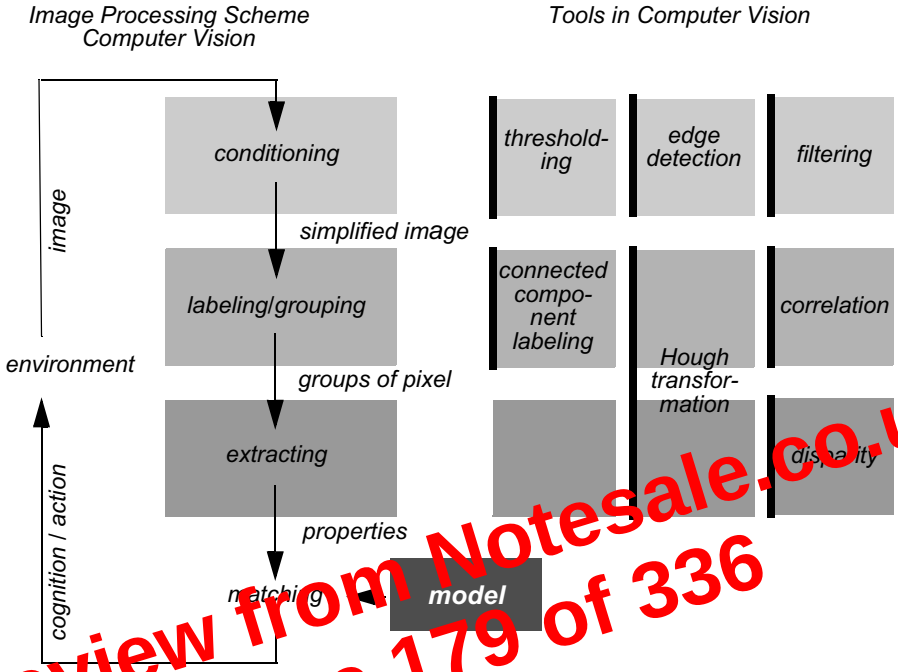


Figure 4.41
Scheme and tools in computer vision. See also [18].

This section presents some appearance-based feature extraction techniques that are relevant to mobile robotics along these lines. Two key requirements must be met for a vision-based feature extraction technique to have mobile robotic relevance. First, the method must operate in real time. Mobile robots move through their environment, and so the processing simply cannot be an off-line operation. Second, the method must be robust to the real-world conditions outside of a laboratory. This means that carefully controlled illumination assumptions and carefully painted objects are unacceptable requirements.

Throughout the following descriptions, keep in mind that vision-based interpretation is primarily about the challenge of *reducing information*. A sonar unit produces perhaps fifty bits of information per second. By contrast, a CCD camera can output 240 *million* bits per second! The sonar produces a tiny amount of information from which we hope to draw broader conclusions. But the CCD chip produces too much information, and this overabundance of information mixes together relevant and irrelevant information haphazardly. For example, we may intend to measure the color of a landmark. The CCD camera does not simply report its color, but also measures the general illumination of the environment, the

In Shakey's environment, edges corresponded to nonfloor objects, and so the floor plane extraction algorithm simply consisted of the application of an edge detector to the monochrome camera image. The lowest edges detected in an image corresponded to the closest obstacles, and the direction of straight-line edges extracted from the image provided clues regarding not only the position but also the orientation of walls and polygonal obstacles.

Although this very simple appearance-based obstacle detection system was successful, it should be noted that special care had to be taken at the time to create indirect lighting in the laboratory such that shadows were not cast, as the system would falsely interpret the edges of shadows as obstacles.

Adaptive floor plane extraction. Floor plane extraction has succeeded not only in artificial environments but in real-world mobile robot demonstrations in which a robot avoids both static obstacles such as walls and dynamic obstacles such as passersby, based on segmentation of the floor plane at a rate of several hertz. Such floor plane extraction algorithms tend to use edge detection and color detection jointly while making certain assumptions regarding the floor, for example, the floor's maximum texture or approximate color range [78].

Each system based on fixed assumptions regarding the floor's appearance is limited to only those environments satisfying its constraints. A more recent approach is that of adaptive floor plane extraction, whereby the parameters defining the expected appearance of the floor are allowed to vary over time. In the simplest instance, one can assume that the pixels at the bottom of the image (i.e., closest to the robot) are part of the floor and contain no obstacles. Then, statistics computed on these "floor sample" pixels can be used to classify the remaining image pixels.

The key challenge in adaptive systems is the choice of what statistics to compute using the "floor sample" pixels. The most popular solution is to construct one or more *histograms* based on the floor sample pixel values. Under "edge detection" above, we found histograms to be useful in determining the best cut point in edge detection thresholding algorithms. Histograms are also useful as discrete representations of distributions. Unlike the Gaussian representation, a histogram can capture multi-modal distributions. Histograms can also be updated very quickly and use very little processor memory. An intensity histogram of the "floor sample" subregion I_f of image I is constructed as follows:

- As preprocessing, smooth I_f, using a Gaussian smoothing operator.
- Initialize a histogram array H with n intensity values: $H[i] = 0$ for $i = 1, \ldots, n$.
- For every pixel (x, y) in I_f increment the histogram: $H[I_f(x, y)] \mathrel{+}= 1$.

The histogram array H serves as a characterization of the appearance of the floor plane. Often, several 1D histograms are constructed, corresponding to intensity, hue, and saturation, for example. Classification of each pixel in I as floor plane or obstacle is performed

sequences generated using the above algorithm. The top string should match *Place 1*, but note that there are deletions and insertions between the two strings.

The technique used in the fingerprinting approach for string differencing is known as a *minimum energy algorithm*. Taken from the stereo vision community, this optimization-based algorithm will find the minimum energy required to "transform" one sequence into another sequence. The result is a distance metric that is relatively insensitive to the addition or subtraction of individual local features while still able to robustly identify the correct matching string in a variety of circumstances.

It should be clear from the previous two sections that whole-image feature extraction is straightforward with vision-based perception and can be applicable to mobile robot localization. But it is spatially localized features that continue to play a dominant role because of their immediate application to the more urgent need for real-time obstacle avoidance.

- Unequal floor contact (slipping, nonplanar surface, etc.).

Some of the errors might be *deterministic (systematic)*, thus they can be eliminated by proper calibration of the system. However, there are still a number of *nondeterministic (random)* errors which remain, leading to uncertainties in position estimation over time. From a geometric point of view one can classify the errors into three types:

1. Range error: integrated path length (distance) of the robot's movement
   → sum of the wheel movements
2. Turn error: similar to range error, but for turns
   → difference of the wheel motions
3. Drift error: difference in the error of the wheels leads to an error in the robot's angular orientation

Over long periods of time, turn and drift errors far outweigh range errors since their contribution to the overall position error is nonlinear. Consider a robot whose position is initially perfectly well-known, moving forward in a straight line along the x-axis. The error in the y-position introduced by a move of d meters will have a component of $d\sin\Delta\theta$, which can be quite large as the angular error $\Delta\theta$ grows. Over time, as a mobile robot moves about the environment, the rotational error between its internal reference frame and its original reference frame grows quickly. As the robot moves away from the origin of these reference frames, the resulting linear error in position grows quite large. It is instructive to establish an error model for odometric accuracy and see how the errors propagate over time.

5.2.4 An error model for odometric position estimation

Generally the pose (position) of a robot is represented by the vector

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5.1}$$

For a differential-drive robot the position can be estimated starting from a known position by integrating the movement (summing the incremental travel distances). For a discrete system with a fixed sampling interval Δt the incremental travel distances $(\Delta x;\Delta y;\Delta\theta)$ are

$$\Delta x = \Delta s\cos(\theta + \Delta\theta/2) \tag{5.2}$$

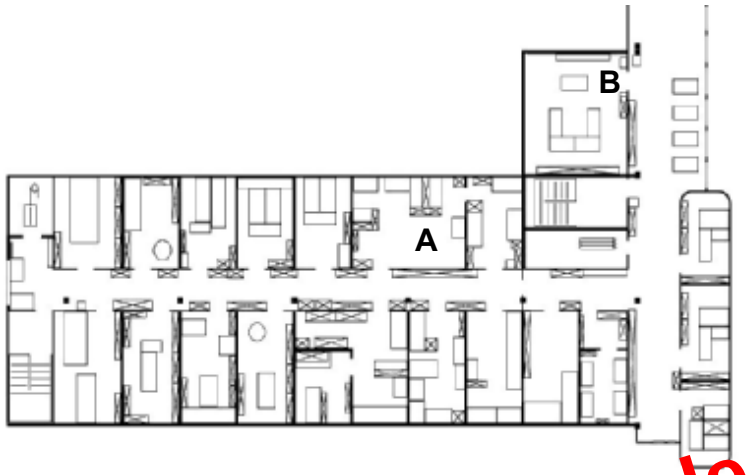


Figure 5.6
A sample environment.

This technique is based on a belief that there exists a procedural solution to the particular navigation problem at hand. For example, in figure 5.6, the behavioralist approach to navigating from room *A* to room *B* might be to design a left-wall following behavior and a detector for room *B* that is triggered by some unique queue in room *B*, such as the color of the carpet. Then the robot can reach room *B* by engaging the left-wall follower with the room *B* detector as the termination condition for the program.

The architecture of this solution to a specific navigation problem is shown in figure 5.7. The key advantage of this method is that, when possible, it may be implemented very quickly for a single environment with a small number of goal positions. It suffers from some disadvantages, however. First, the method does not directly scale to other environments or to larger environments. Often, the navigation code is location-specific, and the same degree of coding and debugging is required to move the robot to a new environment.

Second, the underlying procedures, such as *left-wall-follow,* must be carefully designed to produce the desired behavior. This task may be time-consuming and is heavily dependent on the specific robot hardware and environmental characteristics.

Third, a behavior-based system may have multiple active behaviors at any one time. Even when individual behaviors are tuned to optimize performance, this fusion and rapid switching between multiple behaviors can negate that fine-tuning. Often, the addition of each new incremental behavior forces the robot designer to retune all of the existing behaviors again to ensure that the new interactions with the freshly introduced behavior are all stable.

- The map, if created by the robot, can be used by humans as well, achieving two uses.

The map-based approach will require more up-front development effort to create a navigating mobile robot. The hope is that the development effort results in an architecture that can successfully map and navigate a variety of environments, thereby amortizing the up-front design cost over time.

Of course the key risk of the map-based approach is that an internal representation, rather than the real world itself, is being constructed and *trusted* by the robot. If that model diverges from reality (i.e., if the map is wrong), then the robot's behavior may be undesirable, even if the raw sensor values of the robot are only transiently incorrect.

In the remainder of this chapter, we focus on a discussion of map-based approaches and, specifically, the localization component of these techniques. These approaches are particularly appropriate for study given their significant recent successes in enabling mobile robots to navigate a variety of environments, from academic research buildings, to factory floors, and to museums around the world.

5.4 Belief Representation

The fundamental issue that differentiates various map-based localization systems is the issue of *representation*. There are two specific concepts that the robot must represent, and each has its own unique possible solutions. The robot must have a representation (a model) of the environment—a map. What aspects of the environment are contained in this map? At what level of fidelity does the map represent the environment? These are the design questions for *map representation*.

The robot must also have a representation of its belief regarding its position on the map. Does the robot identify a single unique position as its current position, or does it describe its position in terms of a set of possible positions? If multiple possible positions are expressed in a single belief, how are those multiple positions ranked, if at all? These are the design questions for *belief representation*.

Decisions along these two design axes can result in varying levels of architectural complexity, computational complexity, and overall localization accuracy. We begin by discussing belief representation. The first major branch in a taxonomy of belief representation systems differentiates between single-hypothesis and multiple-hypothesis belief systems. The former covers solutions in which the robot postulates its unique position, whereas the latter enables a mobile robot to describe the degree to which it is uncertain about its position. A sampling of different belief and map representations is shown in figure 5.9.

5.4.1 Single-hypothesis belief

The single-hypothesis belief representation is the most direct possible postulation of mobile robot position. Given some environmental map, the robot's belief about position is

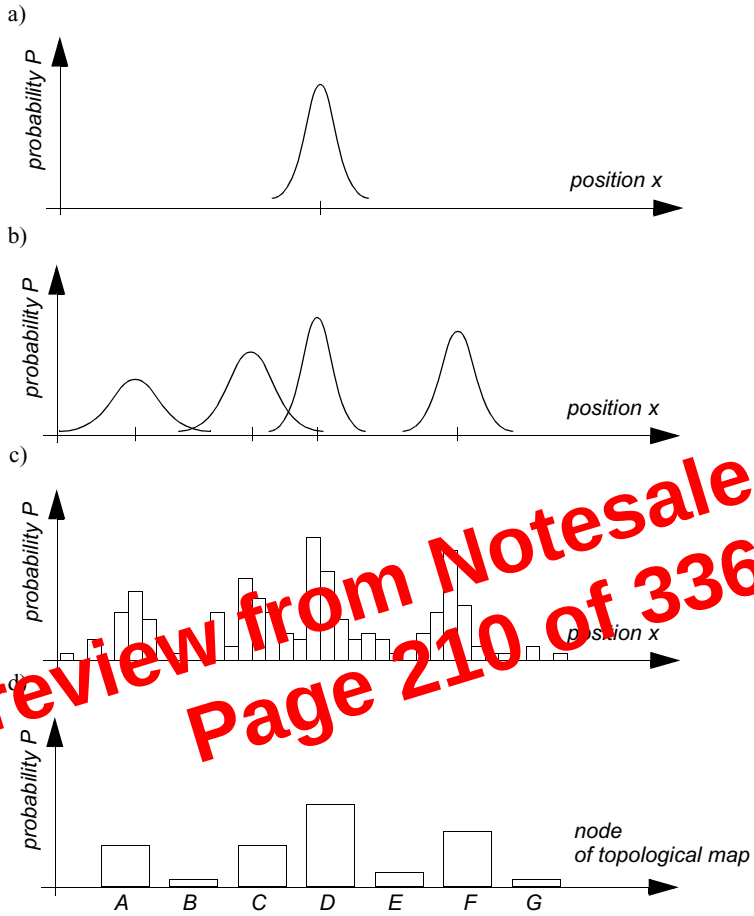


Figure 5.9
Belief representation regarding the robot position (1D) in continuous and discretized (tessellated) maps. (a) Continuous map with single-hypothesis belief, e.g., single Gaussian centered at a single continuous value. (b) Continuous map with multiple-hypothesis belief, e.g;. multiple Gaussians centered at multiple continuous values. (c) Discretized (decomposed) grid map with probability values for all possible robot positions, e.g.; Markov approach. (d) Discretized topological map with probability value for all possible nodes (topological robot positions), e.g.; Markov approach.

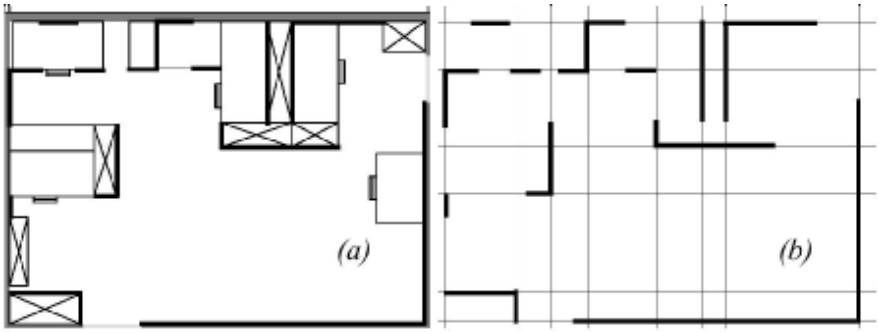


Figure 5.13
Example of a continuous-valued line representation of EPFL. (a) Real map. (b) Representation with a set of infinite lines.

One excellent example involves line extraction. Many indoor mobile robots rely upon laser rangefinding devices to recover distance readings to nearby objects. Such robots can automatically extract best-fit lines from the dense range data provided by thousands of points of laser returns. Given such a line extraction sensor, an appropriate continuous mapping approach is to populate the map with a set of infinite lines. The continuous nature of the map guarantees that lines can be positioned at arbitrary positions in the plane and at arbitrary angles. The abstraction of real environmental objects such as walls and intersections captures only the information in the map representation that matches the type of information recovered by the mobile robot's rangefinding sensor.

Figure 5.13 shows a map of an indoor environment at EPFL using a continuous line representation. Note that the only environmental features captured by the map are straight lines, such as those found at corners and along walls. This represents not only a sampling of the real world of richer features but also a simplification, for an actual wall may have texture and relief that is not captured by the mapped line.

The impact of continuous map representations on position representation is primarily positive. In the case of single-hypothesis position representation, that position may be specified as any continuous-valued point in the coordinate space, and therefore extremely high accuracy is possible. In the case of multiple-hypothesis position representation, the continuous map enables two types of multiple position representation.

In one case, the possible robot position may be depicted as a geometric shape in the hyperplane, such that the robot is known to be within the bounds of that shape. This is shown in figure 5.29, in which the position of the robot is depicted by an oval bounding area.

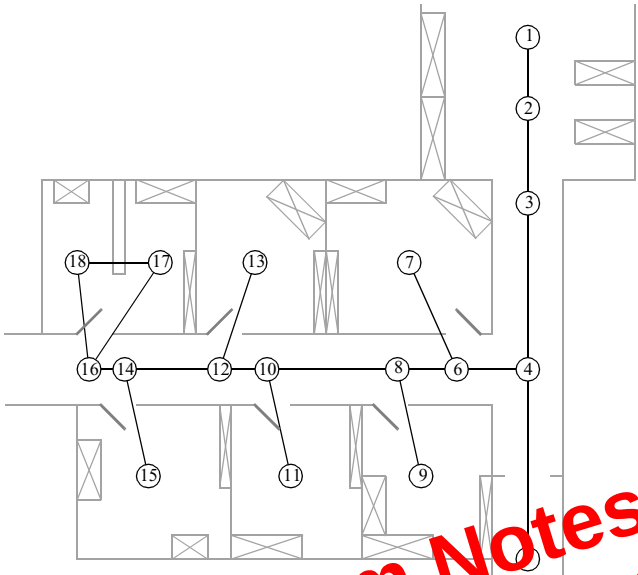


Figure 5.13
A topological representation of an indoor office area.

rooms. Note that nodes capture geometric space, and arcs in this representation simply represent connectivity.

Another example of topological representation is the work of Simhon and Dudek [134], in which the goal is to create a mobile robot that can capture the most interesting aspects of an area for human consumption. The nodes in their representation are visually striking locales rather than route intersections.

In order to navigate using a topological map robustly, a robot must satisfy two constraints. First, it must have a means for detecting its current position in terms of the nodes of the topological graph. Second, it must have a means for traveling between nodes using robot motion. The node sizes and particular dimensions must be optimized to match the sensory discrimination of the mobile robot hardware. This ability to "tune" the representation to the robot's particular sensors can be an important advantage of the topological approach. However, as the map representation drifts further away from true geometry, the expressiveness of the representation for accurately and precisely describing a robot position is lost. Therein lies the compromise between the discrete cell-based map representations and the topological representations. Interestingly, the continuous map representation has

5.6.2.1 Introduction: applying probability theory to robot localization

Given a discrete representation of robot positions, in order to express a belief state we wish to assign to each possible robot position a probability that the robot is indeed at that position. From probability theory we use the term $p(A)$ to denote the probability that A is true. This is also called the *prior probability* of A because it measures the probability that A is true independent of any additional knowledge we may have. For example we can use $p(r_t = l)$ to denote the prior probability that the robot r is at position l at time t.

In practice, we wish to compute the probability of each individual robot position given the encoder and sensor evidence the robot has collected. In probability theory, we use the term $p(A|B)$ to denote the *conditional* probability of A given that we know B. For example, we use $p(r_t = l|i_t)$ to denote the probability that the robot is at position l given that the robot's sensor inputs i.

The question is, how can a term such as $p(r_t = l|i_t)$ be simplified to its constituent parts so that it can be computed? The answer lies in the product rule, which states

$$p(A \wedge B) = p(A|B)p(B) \tag{5.18}$$

Equation (5.18) is intuitively straightforward, as the probability of both A and B being true is being related to A being true *and* the other being conditionally true. But you should be able to convince yourself that the alternate equation is equally correct:

$$p(A \wedge B) = p(B|A)p(A) \tag{5.19}$$

Using equations (5.18) and (5.19) together, we can derive the Bayes formula for computing $p(A|B)$:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{5.20}$$

We use the Bayes rule to compute the robot's new belief state as a function of its sensory inputs and its former belief state. But to do this properly, we must recall the basic goal of the Markov localization approach: a discrete set of possible robot positions L are represented. The belief state of the robot must assign a probability $p(r_t = l)$ for each location l in L.

The *See* function described in equation (5.17) expresses a mapping from a belief state and sensor input to a refined belief state. To do this, we must update the probability associated with each position l in L, and we can do this by directly applying the Bayes formula to every such l. In denoting this, we will stop representing the temporal index t for simplicity and will further use $p(l)$ to mean $p(r = l)$:

each position. An example of just such a procedure is the sensory uncertainty field of Latombe [141], in which the robot must find a trajectory that reaches its goal while maximizing its localization confidence on-line.

5.6.2.3 Case study 2: Markov localization using a grid map

The major weakness of a purely topological decomposition of the environment is the resolution limitation imposed by such a granular representation. The position of the robot is usually limited to the resolution of a single node in such cases, and this may be undesirable for certain applications.

In this case study, we examine the work of Burgard and colleagues [49, 50] in which far more precise navigation is made possible using a grid-based representation while still employing the Markov localization technique.

The robot used by this research, Rhino, is an RWI B24 robot with twenty-four sonars and two Sick laser rangefinders. Clearly, at the sensory level this robot accumulates greater and more accurate range data than is possible with the handful of sonar sensors mounted on Dervish. In order to make maximal use of these fine-grained sensory data, Rhino uses a 2D geometric environmental representation of free and occupied space. This metric map is tessellated regularly into a *fixed decomposition* grid with each cell occupying 4 to 64 cm in various instantiations.

Like Dervish, Rhino uses multiple-hypothesis belief representation. In line with the far improved resolution of the environmental representation, the belief state representation of Rhino consists of $15 \times 15 \times 15$ 3D array representing the probability of 15^3 possible robot positions (see figure 5.23). The resolution of the array is $15\ \text{cm} \times 15\ \text{cm} \times 1^\circ$. Note that unlike Dervish, which assumes its orientation is approximate and known, Rhino explicitly represents fine-grained alternative orientations, and so its belief state formally represents three degrees of freedom. As we have stated before, the resolution of the belief state representation must match the environmental representation in order for the overall system to function well.

Whereas Dervish made use of only perceptual events, ignoring encoder inputs and therefore metric distance altogether, Rhino uses the complete Markov probabilistic localization approach summarized in section 5.6.2.1, including both an explicit action update phase and a perception update phase at every cycle.

The discrete Markov chain version of action update is performed because of the tessellated representation of position. Given encoder measurements o at time t, each updated position probability in the belief state is expressed as a sum over previous possible positions and the motion model:

$$P(l_t|o_t) = \sum_{l'} P(l_t|l'_{t-1}, o_t) \cdot p(l'_{t-1}) \tag{5.26}$$

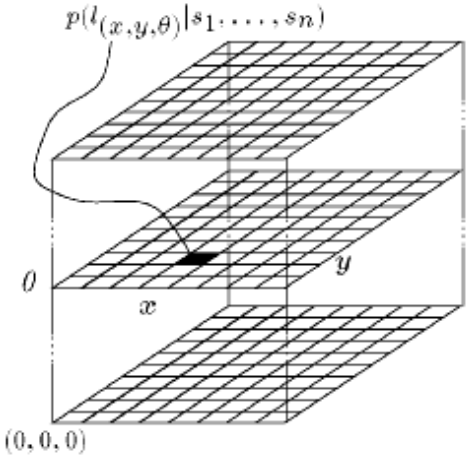


Figure 5.23
The belief state representation 3D array used by Rhino (courtesy of W. Burgard and S. Thrun).

Note that equation (5.26) is simply a discrete version of equation (5.22). The specific motion model used by Rhino represents the result of motion as a Gaussian that is bounded (i.e., the tails of the distribution are finite). Rhino's kinematic configuration is a three-wheel synchro-drive rather than a differential-drive robot. Nevertheless, the error ellipses depicted in figures 5.4 and 5.5 are similar to the Gaussian bounds that result from Rhino's motion model.

The perception model follows the Bayes formula precisely, as in equation (5.21). Given a range perception i the probability of the robot being at each location l is updated as follows:

$$p(l|i) = \frac{p(i|l)p(l)}{p(i)} \tag{5.27}$$

Note that a denominator is used by Rhino, although the denominator is constant for varying values of l. This denominator acts as a normalizer to ensure that the probability measures in the belief state continue to sum to 1.

The critical challenge is, of course, the calculation of $p(i|l)$. In the case of Dervish, the number of possible values for i and l were so small that a simple table could suffice. However, with the fine-grained metric representation of Rhino, the number of possible sensor readings and environmental geometric contexts is extremely large. Thus, Rhino computes

$$\sum_{i=1}^{n} w_i \hat{q} - \sum_{i=1}^{n} w_i q_i = 0 \tag{5.32}$$

$$\hat{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i} \tag{5.33}$$

If we take as the weight w_i

$$w_i = \frac{1}{\sigma_i^2} \tag{5.34}$$

then the value of $\hat{q}$ in terms of two measurements can be defined as follows:

$$\hat{q} = \frac{\frac{1}{\sigma_1^2} q_1 + \frac{1}{\sigma_2^2} q_2}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} q_2 \tag{5.35}$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} = \frac{\sigma_2^2 + \sigma_1^2}{\sigma_1^2 \sigma_2^2} \quad ; \quad \sigma^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_2^2 + \sigma_1^2} \tag{5.36}$$

Note that from equation (5.36) we can see that the resulting variance σ^2 is less than all the variances σ_i^2 of the individual measurements. Thus the uncertainty of the position estimate has been decreased by combining the two measurements. The solid probability density curve represents the result of the Kalman filter in figure 5.26, depicting this result. Even poor measurements, such as are provided by the sonar, will only increase the precision of an estimate. This is a result that we expect based on information theory.

Equation (5.35) can be rewritten as

$$\hat{q} = q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}(q_2 - q_1) \tag{5.37}$$

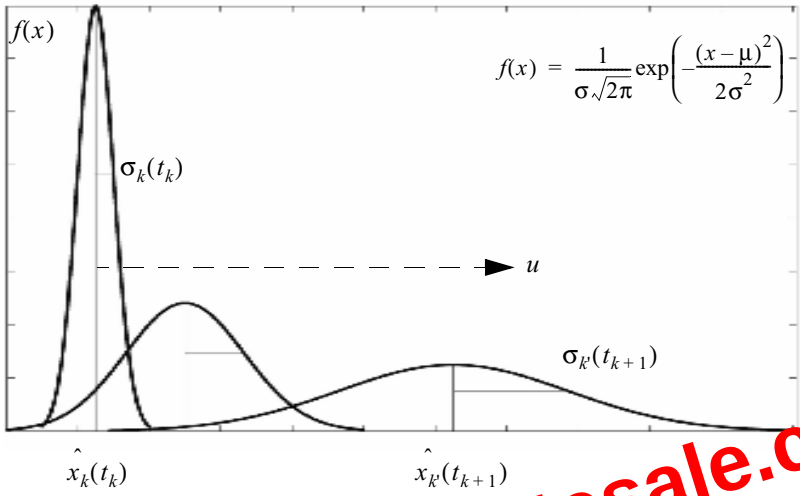


Figure 5.27
Propagation of probability density of a moving robot [1706].

The optimal estimate at time $k + 1$ is given by the last estimate at k and the estimate of the robot motion including the estimated movement errors.

By extending the above equations to the vector case and allowing time-varying parameters in the system and a description of noise, we can derive the Kalman filter localization algorithm.

5.6.3.2 Application to mobile robots: Kalman filter localization

The Kalman filter is an optimal and efficient sensor fusion technique. Application of the Kalman filter to localization requires posing the robot localization problem as a sensor fusion problem. Recall that the basic probabilistic update of robot belief state can be segmented into two phases, *perception update* and *action update,* as specified by equations (5.21) and (5.22).

The key difference between the Kalman filter approach and our earlier Markov localization approach lies in the perception update process. In Markov localization, the entire perception, that is, the robot's set of instantaneous sensor measurements, is used to update each possible robot position in the belief state individually using the Bayes formula. In some cases, the perception is abstract, having been produced by a feature extraction mechanism, as in Dervish. In other cases, as with Rhino, the perception consists of raw sensor readings.

By contrast, perception update using a Kalman filter is a multistep process. The robot's total sensory input is treated not as a monolithic whole but as a set of extracted features that

$$\hat{Z}(k+1) = \{\hat{z}_i(k+1) | (1 \le i \le n_t)\} \tag{5.49}$$

The predicted state estimate $\hat{p}(k+1|k)$ is used to compute the measurement Jacobian ∇h_i for each prediction. As you will see in the example below, the function h_i is mainly a coordinate transformation between the world frame and the sensor frame.

4. Matching. At this point we have a set of actual, single observations, which are features in sensor space, and we also have a set of predicted features, also positioned in sensor space. The matching step has the purpose of identifying all of the single observations that match specific predicted features well enough to be used during the estimation process. In other words, we will, for a subset of the observations and a subset of the predicted features, find pairings that intuitively say "this observation is the robot's measurement of this predicted feature based on the map."

Formally, the goal of the matching procedure is to produce an assignment from observations $z_j(k+1)$ to the targets z_t (stored in the map). For each measurement prediction for which a corresponding observation is found we calculate the innovation $v_{ij}(k+1)$. *Innovation* is a measure of the difference between the predicted and observed measurements:

$$\begin{aligned} v_{ij}(k+1) &= [z_j(k+1) - \hat{z}_i(k+1)] \\ &= [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))] \end{aligned} \tag{5.50}$$

The innovation covariance $\Sigma_{IN,ij}(k+1)$ can be found by applying the error propagation law [section 4.2.2, equation (4.60)]:

$$\Sigma_{IN,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^T + \Sigma_{R,i}(k+1) \tag{5.51}$$

where $\Sigma_{R,i}(k+1)$ represents the covariance (noise) of the measurement $z_i(k+1)$.

To determine the validity of the correspondence between measurement prediction and observation, a *validation gate* has to be specified. A possible definition of the validation gate is the Mahalanobis distance:

$$v_{ij}^T(k+1) \cdot \Sigma_{IN,ij}^{-1}(k+1) \cdot v_{ij}(k+1) \le g^2 \tag{5.52}$$

However, dependent on the application, the sensors, and the environment models, more sophisticated validation gates might be employed.

The validation equation is used to test observation $z_j(k+1)$ for membership in the validation gate for each predicted measurement. When a single observation falls in the validation gate, we get a successful match. If one observation falls in multiple validation gates,

the best matching candidate is selected or multiple hypotheses are tracked. Observations that do not fall in the validation gate are simply ignored for localization. Such observations could have resulted from objects not in the map, such as new objects (e.g., someone places a large box in the hallway) or transient objects (e.g., humans standing next to the robot may form a line feature). One approach is to take advantage of such unmatched observations to populate the robot's map.

5. Estimation: applying the Kalman filter. Next we compute the best estimate $\hat{p}(k+1|k+1)$ of the robot's position based on the position prediction and all the observations at time $k+1$. To do this position update, we first stack the validated observations $z_j(k+1)$ into a single vector to form $z(k+1)$ and designate the composite innovation $v(k+1)$. Then we stack the measurement Jacobians ∇h_i for each validated measurement together to form the composite Jacobian ∇h and the measurement error (noise) vector $\Sigma_R(k+1) = diag[\Sigma_{R,i}(k+1)]$. We can then compute the composite innovation covariance $\Sigma_{IN}(k+1)$ according to equation (5.51) and by utilizing the well-known result [3] that the Kalman gain can be written as

$$K(k+1) = \Sigma_p(k+1|k) \cdot \nabla h^T \cdot \Sigma_{IN}^{-1}(k+1) \tag{5.53}$$

we can update the robot's position estimate

$$\hat{p}(k+1|k+1) = p(k+1|k) + K(k+1) \cdot v(k+1) \tag{5.54}$$

with the associated variance

$$\Sigma_p(k+1|k+1) = \Sigma_p(k+1|k) - K(k+1) \cdot \Sigma_{IN}(k+1) \cdot K^T(k+1) \tag{5.55}$$

For the 1D case and with $h_i(z_t, \hat{p}(k+1|k)) = z_t$ we can show that this formula corresponds to the 1D case derived earlier

Equation (5.53) is simplified to

$$K(k+1) = \frac{\sigma_p^2(k+1|k)}{\sigma_{IN}^2(k+1)} = \frac{\sigma_p^2(k+1|k)}{\sigma_p^2(k+1|k) + \sigma_R^2(k+1)} \tag{5.56}$$

corresponding to equation (5.45), and equation (5.54) simplifies to

that the robot will always be able to localize successfully. This work also led to a real-world demonstration of landmark-based localization. Standard sheets of paper were placed on the ceiling of the Robotics Laboratory at Stanford University, each with a unique checkerboard pattern. A Nomadics 200 mobile robot was fitted with a monochrome CCD camera aimed vertically up at the ceiling. By recognizing the paper landmarks, which were placed approximately 2 m apart, the robot was able to localize to within several centimeters, then move, using dead reckoning, to another landmark zone.

The primary disadvantage of landmark-based navigation is that in general it requires significant environmental modification. Landmarks are local, and therefore a large number are usually required to cover a large factory area or research laboratory. For example, the Robotics Laboratory at Stanford made use of approximately thirty discrete landmarks, all affixed individually to the ceiling.

5.7.2 Globally unique localization

The landmark-based navigation approach makes a strong general assumption: when the landmark is in the robot's field of view, localization is essentially perfect. One way to reach the Holy Grail of mobile robotic localization is to effectively enable such an assumption to be valid no matter *where* the robot is located. It would be revolutionary if a look at the robot's sensors immediately identified its particular location, uniquely and repeatedly.

Such a strategy for localization is surely aggressive, but the question of whether it can be done is primarily a question of sensor technology and sensing software. Clearly, such a localization system would need to use a sensor that collects a very large amount of information. Since vision does indeed collect far more information than previous sensors, it has been used as the sensor of choice in research toward globally unique localization.

Figure 4.49 depicts the image taken by a catadioptric camera system. If humans were able to look at an individual such picture and identify the robot's location in a well-known environment, then one could argue that the information for globally unique localization does exist within the picture; it must simply be teased out.

One such approach has been attempted by several researchers and involves constructing one or more image histograms to represent the information content of an image stably (see e.g., figure 4.50 and section 4.3.2.2). A robot using such an image-histogramming system has been shown to uniquely identify individual rooms in an office building as well as individual sidewalks in an outdoor environment. However, such a system is highly sensitive to external illumination and provides only a level of localization resolution equal to the visual footprint of the camera optics.

The angular histogram depicted in figure 4.39 of the previous chapter is another example in which the robot's sensor values are transformed into an identifier of location. However, due to the limited information content of sonar ranging strikes, it is likely that two *places*

5.8 Autonomous Map Building

All of the localization strategies we have discussed require human effort to install the robot into a space. Artificial environmental modifications may be necessary. Even if this not be case, a map of the environment must be created for the robot. But a robot that localizes successfully has the right sensors for detecting the environment, and so the robot ought to build its own map. This ambition goes to the heart of autonomous mobile robotics. In prose, we can express our eventual goal as follows:

Starting from an arbitrary initial point, a mobile robot should be able to autonomously explore the environment with its on-board sensors, gain knowledge about it, interpret the scene, build an appropriate map, and localize itself relative to this map.

Accomplishing this goal robustly is probably years away, but an important subgoal is the invention of techniques for autonomous creation and modification of an environmental map. Of course a mobile robot's sensors have only a limited range, and so it must physically explore its environment to build such a map. So, the robot must not only create a map but it must do so while moving and localizing to explore the environment. In the robotics community, this is often called the simultaneous localization and mapping (SLAM) problem, arguably the most difficult problem specific to mobile robot systems.

The reason that SLAM is difficult is born precisely from the interaction between the robot's position updates as it localizes and its mapping actions. If a mobile robot updates its position based on an observation of an imprecisely known feature, the resulting position estimate becomes correlated with the feature location estimate. Similarly, the map becomes correlated with the position estimate if an observation taken from an imprecisely known position is used to update or add a feature to the map. The general problem of map-building is thus an example of the chicken-and-egg problem. For localization the robot needs to know where the features are, whereas for map-building the robot needs to know where it is on the map.

The only path to a complete and optimal solution to this joint problem is to consider all the correlations between position estimation and feature location estimation. Such cross-correlated maps are called *stochastic* maps, and we begin with a discussion of the theory behind this approach in the following section [55].

Unfortunately, implementing such an optimal solution is computationally prohibitive. In response a number of researchers have offered other solutions that have functioned well in limited circumstances. Section 5.8.2 characterizes these alternative partial solutions.

5.8.1 The stochastic map technique

Figure 5.38 shows a general schematic incorporating map building and maintenance into the standard localization loop depicted by figure 5.28 during the discussion of Kalman filter

features and avoids a great deal of irrelevant detail. When the robot arrives at a topological node that could be the same as a previously visited and mapped node (e.g., similar distinguishing features), then the robot postulates that it has indeed returned to the same node. To check this hypothesis, the robot explicitly plans and moves to adjacent nodes to see if its perceptual readings are consistent with the cycle hypothesis.

With the recent popularity of metric maps, such as fixed decomposition grid representations, the cycle detection strategy is not as straightforward. Two important features are found in most autonomous mapping systems that claim to solve the cycle detection problem. First, as with many recent systems, these mobile robots tend to accumulate recent perceptual history to create small-scale local *submaps* [51, 74, 157]. Each submap is treated as a single sensor during the robot's position update. The advantage of this approach is twofold. Because odometry is relatively accurate over small distances, the relative registration of features and raw sensor strikes in a local submap will be quite accurate. In addition to this, the robot will have created a virtual sensor system with a significantly larger horizon than its actual sensor system's range. In a sense, this strategy at the very least defers the problem of very large cyclic environments by increasing the map scale that can be handled well by the robot.

The second recent technique for dealing with cycle environments is in fact a return to the topological representation. Some recent automatic mapping systems will attempt to identify cycles by associating a topology with the set of metric submaps, explicitly identifying the loops first at the topological level. In the case of [51], for example, the topological level loop is identified with a human who pushes a button at a known landmark position. In the case of [74], the topological level loop is determined by performing correspondence tests between submaps, postulating that two submaps represent the same place in the environment when the correspondence is good.

One could certainly imagine other augmentations based on known topological methods. For example, the globally unique localization methods described in section 5.7 could be used to identify topological correctness. It is notable that the automatic mapping research of the present has, in many ways, returned to the basic topological correctness question that was at the heart of some of the earliest automatic mapping research in mobile robotics more than a decade ago. Of course, unlike that early work, today's automatic mapping results boast correct cycle detection combined with high-fidelity geometric maps of the environment.

5.8.2.2 Dynamic environments

A second challenge extends not just to existing autonomous mapping solutions but to the basic formulation of the stochastic map approach. All of these strategies tend to assume that the environment is either unchanging or changes in ways that are virtually insignificant. Such assumptions are certainly valid with respect to some environments, such as, for example, the computer science department of a university at 3 AM. However, in a great many

6.2 Competences for Navigation: Planning and Reacting

In the artificial intelligence community planning and reacting are often viewed as contrary approaches or even opposites. In fact, when applied to physical systems such as mobile robots, planning and reacting have a strong complementarity, each being critical to the other's success. The navigation challenge for a robot involves executing a course of action (or plan) to reach its goal position. During execution, the robot must react to unforeseen events (e.g., obstacles) in such a way as to still reach the goal. Without reacting, the planning effort will not pay off because the robot will never physically reach its goal. Without planning, the reacting effort cannot guide the overall robot behavior to reach a distant goal – again, the robot will never reach its goal.

An information-theoretic formulation of the navigation problem will make this complementarity clear. Suppose that a robot M at time i has a map M_i and an initial belief state b_i. The robot's goal is to reach a position p while satisfying some temporal constraints: $loc_g(R) = p \; ; (g \leq n)$. Thus the robot must be at location p at or before timestep n.

Although the goal of the robot is distinctly physical, the robot can only really sense its belief state, not its physical location, and therefore we map the goal of reaching location p to reaching a belief state b_g corresponding to the belief that $loc_g(R) = p$. With this formulation, a plan q is nothing more than one or more trajectories from b_i to b_g. In other words, plan q will cause the robot's belief state to transition from b_i to b_g if the plan is executed from a world state consistent with both b_i and M_i.

Of course the problem is that the latter condition may not be met. It is entirely possible that the robot's position is not quite consistent with b_i, and it is even likelier that M_i is either incomplete or incorrect. Furthermore, the real-world environment is dynamic. Even if M_i is correct as a single snapshot in time, the planner's model regarding how M changes over time is usually imperfect.

In order to reach its goal nonetheless, the robot must incorporate new information gained during plan execution. As time marches forward, the environment changes and the robot's sensors gather new information. This is precisely where reacting becomes relevant. In the best of cases, reacting will modulate robot behavior locally in order to correct the planned-upon trajectory so that the robot still reaches the goal. At times, unanticipated new information will require changes to the robot's strategic plans, and so ideally the planner also incorporates new information as that new information is received.

Taken to the limit, the planner would incorporate every new piece of information in real time, instantly producing a new plan that in fact reacts to the new information appropriately. This theoretical extreme, at which point the concept of planning and the concept of reacting merge, is called *integrated planning and execution* and is discussed in section 6.3.4.3.

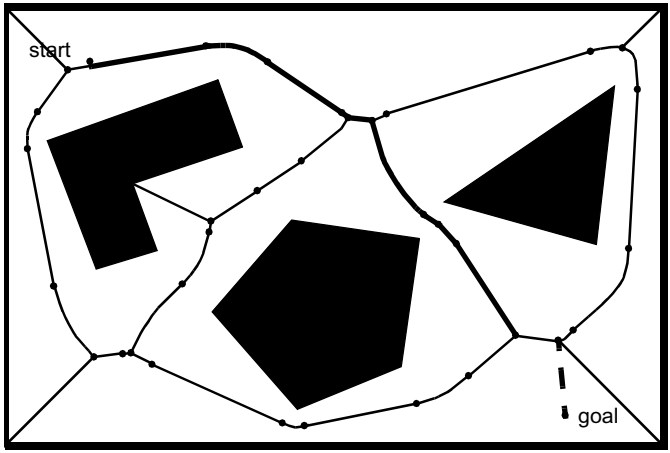


Figure 6.3
Voronoi diagram [21]. The Voronoi diagram consists of the lines constructed from all points that are equidistant from two or more obstacles. The initial q_{init} and goal q_{goal} configurations are mapped into the Voronoi diagram to q'_{init} and q'_{goal}, each by drawing the line along which its distance to the boundary of the obstacle increases the fastest. The direction of movement on the Voronoi diagram is also selected so that the distance to the boundaries increases fastest. The points on the Voronoi diagram represent transitions from line segments (minimum distance between two lines) to parabolic segments (minimum distance between a line and a point).

to the goal. More formally, we can prove that visibility graph planning is *optimal* in terms of the length of the solution path. This powerful result also means that all sense of safety, in terms of staying a reasonable distance from obstacles, is sacrificed for this optimality. The common solution is to grow obstacles by significantly more than the robot's radius, or, alternatively, to modify the solution path after path planning to distance the path from obstacles when possible. Of course such actions sacrifice the optimal-length results of visibility graph path planning.

Voronoi diagram. Contrasting with the visibility graph approach, a Voronoi diagram is a complete road map method that tends to maximize the distance between the robot and obstacles in the map. For each point in the free space, compute its distance to the nearest obstacle. Plot that distance in figure 6.3 as a height coming out of the page. The height increases as you move away from an obstacle. At points that are equidistant from two or more obstacles, such a distance plot has sharp ridges. The Voronoi diagram consists of the edges formed by these sharp ridge points. When the configuration space obstacles are polygons, the Voronoi diagram consists of straight and parabolic segments. Algorithms that

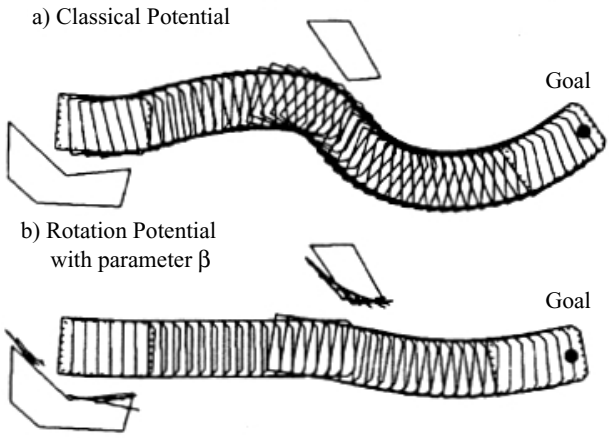


Figure 6.6
Comparison between a classical potential field and an extended potential field. Image courtesy of Raja Chatila [84].

a gain factor which reduces the repulsive force when an obstacle is parallel to the robot's direction of travel, since such an object does not pose an immediate threat to the robot's trajectory. The result is enhanced wall following, which was problematic for earlier implementations of potential fields methods.

The task potential field considers the present robot velocity and from that it filters out those obstacles that should not affect the near-term potential based on robot velocity. Again a scaling is made, this time of all obstacle potentials when there are no obstacles in a sector named Z in front of the robot. The sector Z is defined as the space which the robot will sweep during its next movement. The result can be smoother trajectories through space. An example comparing a classical potential field and an extended potential field is depicted in figure 6.6.

A great many variations and improvements of the potential field methods have been proposed and implemented by mobile roboticists [67, 111]. In most cases, these variations aim to improve the behavior of potential fields in local minima while also lowering the chances of oscillations and instability when a robot must move through a narrow space such as a doorway.

Potential fields are extremely easy to implement, much like the grassfire algorithm described in section 6.2.1.2. Thus it has become a common tool in mobile robot applications in spite of its theoretical limitations.

This completes our brief summary of the path-planning techniques that are most popular in mobile robotics. Of course, as the complexity of a robot increases (e.g., large degree of

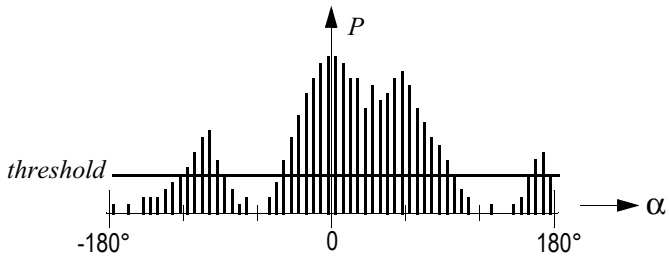


Figure 6.9
Polar histogram [93].

One of the central criticisms of Bug-type algorithms is that the robot's behavior at each instant is generally a function of only its most recent sensor readings. This can lead to undesirable and yet preventable problems in cases where the robot's instantaneous sensor readings do not provide enough information for robust obstacle avoidance. The VFH techniques overcome this limitation by creating a local map of the environment around the robot. This local map is a small occupancy grid, as described in section 5.7, populated only by relatively recent sensor range readings. For obstacle avoidance, VFH generates a polar histogram as shown in figure 6.9. The x-axis represents the angle α at which the obstacle was found and the y-axis represents the probability P that there really is an obstacle in that direction based on the occupancy grid's cell values.

From this histogram a steering direction is calculated. First all openings large enough for the vehicle to pass through are identified. Then a cost function is applied to every such candidate opening. The passage with the lowest cost is chosen. The cost function G has three terms:

$$G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction} \qquad (6.11)$$

target_direction = alignment of the robot path with the goal;

wheel_orientation = difference between the new direction and the current wheel orientation;

previous_direction = difference between the previously selected direction and the new direction.

The terms are calculated such that a large deviation from the goal direction leads to a big cost in the term "target direction". The parameters a, b, c in the cost function G tune the behavior of the robot. For instance, a strong goal bias would be expressed with a large value for a. For a complete definition of the cost function, refer to [92].

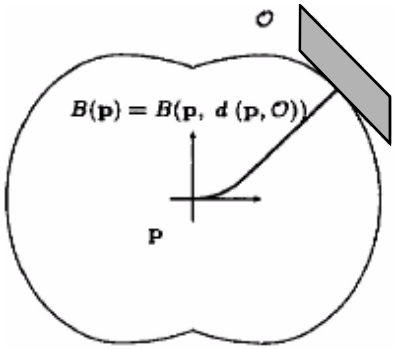


Figure 6.11
Shape of the bubbles around the vehicle (courtesy of Raja Chatila [85]).

vehicles and so we focus only on the bubble band extension made by Khatib, Jaouni, Chatila, and Laumod [85].

A *bubble* is defined as the maximum local subset of the free space around a given configuration of the robot that which can be traveled in any direction without collision. The bubble is generated using a simplified model of the robot in conjunction with range information available in the robot's map. Even with a simplified model of the robot's geometry, it is possible to take into account the actual shape of the robot when calculating the bubble's size (figure 6.11). Given such bubbles, a band or string of bubbles can be used along the trajectory from the robot's initial position to its goal position to show the robot's expected free space throughout its path (see figure 6.12).

Clearly, computing the bubble band requires a global map and a global path planner. Once the path planner's initial trajectory has been computed and the bubble band is calculated, then modification of the planned trajectory ensues. The bubble band takes into account forces from modeled objects and internal forces. These internal forces try to minimize the "slack" (energy) between adjacent bubbles. This process, plus a final smoothing operation, makes the trajectory smooth in the sense that the robot's free space will change as smoothly as possible during path execution.

Of course, so far this is more akin to path optimization than obstacle avoidance. The obstacle avoidance aspect of the bubble band strategy comes into play during robot motion. As the robot encounters unforeseen sensor values, the bubble band model is used to deflect the robot from its originally intended path in a way that minimizes bubble band *tension*.

An advantage of the bubble band technique is that one can account for the actual dimensions of the robot. However, the method is most applicable only when the environment configuration is well-known ahead of time, just as with off-line path-planning techniques.

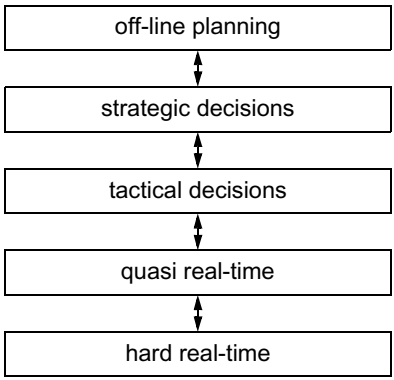


Figure 6.18
Generic temporal decomposition of a navigation architecture.

affects the robot's immediate actions and is therefore subject to some temporal constraints, while a strategic or off-line layer represents decisions that affect the robot's behavior over the long term, with few temporal constraints on the module's response time.

Four important, interrelated trends correlate with temporal decomposition. These are not set in stone; there are exceptions. Nevertheless, these general properties of temporal decompositions are enlightening:

Sensor response time. A particular module's sensor response time can be defined as the amount of time between acquisition of a sensor-based event and a corresponding change in the output of the module. As one moves up the stack in figure 6.18 the sensor response time tends to increase. For the lowest-level modules, the sensor response time is often limited only by the raw processor and sensor speeds. At the highest-level modules, sensor response can be limited by slow and deliberate decision-making processes.

Temporal depth. Temporal depth is a useful concept applying to the temporal window that affects the module's output, both backward and forward in time. *Temporal horizon* describes the amount of look ahead used by the module during the process of choosing an output. *Temporal memory* describes the historical time span of sensor input that is used by the module to determine the next output. Lowest-level modules tend to have very little temporal depth in both directions, whereas the deliberative processes of highest-level modules make use of a large temporal memory and consider actions based on their long-term consequences, making note of large temporal horizons.

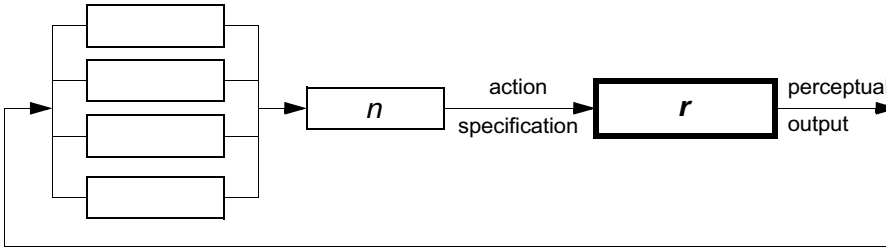


Figure 6.21
Example of a pure parallel decomposition.

Such a serial system uses the internal state of all associated modules and the value of the robot's percept I in a sequential manner to compute the next robot action, O. A pure serial architecture has advantages relating to predictability and verifiability. Since the state and outputs of each module depend entirely on the input it receives from the module upstream, the entire system, including the robot, is a single well-formed loop. Therefore, the overall behavior of the system can be evaluated using well-known discrete forward simulation methods.

Figure 6.21 depicts the extreme opposite of pure serial control, a fully parallel control architecture. Because we choose to define r as a module with precisely one input, this parallel system includes a special module n that provides a single output for the consumption of r. Intuitively, the fully parallel system distributes responsibility for the system's control output O across multiple modules, possibly simultaneously. In a pure sequential system, the control flow is a linear sequence through a string of modules. Here, the control flow contains a *combination* step at which point the result of multiple modules may impact O in arbitrary ways.

Thus parallelization of control leads to an important question: how will the output of each component module inform the overall decision concerning the value of O? One simple combination technique is temporal switching. In this case, called *switched parallel*, the system has a parallel decomposition but at any particular instant in time the output O can be attributed to one specific module. The value of O can of course depend on a different module at each successive time instant, but the instantaneous value of O can always be determined based on the functions of a single module. For instance, suppose that a robot has an obstacle avoidance module and a path-following module. One switched control implementation may involve execution of the path-following recommendation whenever the robot is more than 50 cm from all sensed obstacles and execution of the obstacle avoidance recommendation when any sensor reports a range closer than 50 cm.

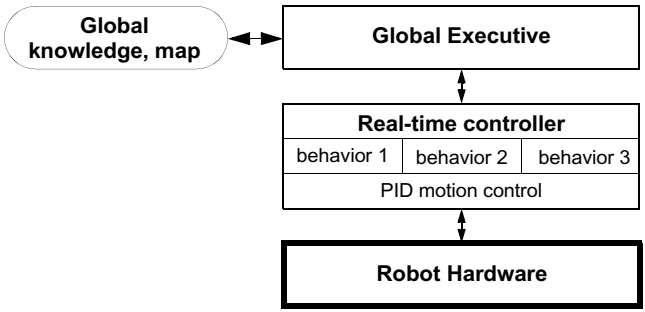


Figure 6.26
An integrated planning and execution architecture in which planning is nothing more than a real-time execution step (behavior).

robot is not kinematically symmetric, and so servoing through a particular obstacle course may be easier in one direction than the other.

Pygmalion's environment representation consists of a continuous geometric model as well as an abstract topological network for route planning. Thus, if repeated attempts to clear the obstacle fail, then the robot's executive will temporarily cut the topological connection between the two appropriate nodes and will launch the planner again, generating a new set of waypoints to the goal. Next, using recent laser rangefinding data as a type of local map (see figure 6.25), a geometric path planner will generate a path from the robot's current position to the next waypoint.

In summary, episodic planning architectures are extremely popular in the mobile robot research community. They combine the versatility of responding to environmental changes and new goals with the fast response of a tactical executive tier and behaviors that control real-time robot motion. As shown in figure 6.25, it is common in such systems to have both a short-term local map and a more strategic global map. Part of the executive's job in such dual representations is to decide when and if new information integrated into the local map is sufficiently nontransient to be copied into the global knowledge base.

6.3.4.3 Integrated planning and execution

Of course, the architecture of a commercial mobile robot must include more functionality than just navigation. But limiting this discussion to the question of *navigation* architectures leads to what may at first seem a degenerate solution.

The architecture shown in figure 6.26 may look similar to the off-line planning architecture of figure 6.24, but in fact it is significantly more advanced. In this case, the planner tier has disappeared because there is no longer a temporal decomposition between the executive

[68] Fox, D., "KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization." *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[69] Fox, D., Burgard,W., Thrun, S., "The Dynamic Window Approach to Collision Avoidance." *IEEE Robotics and Automation Magazine*, 4:23–33, 1997.

[70] Gander,W., Golub, G.H., Strebel, R., "Least-Squares Fitting of Circles and Ellipses." *BIT Numerical Mathematics*, vol. 34, no. 4, pp. 558–578, December 1994.

[71] Genesereth, M.R. "Deliberate Agents." *Technical Report Logic-87-2*. Stanford, CA, Stanford University, Logic Group, 1987.

[72] Golub, G., Kahan,W., "Calculating the Singular Values and Pseudo-Inverse of a Matrix." *Journal SIAM Numerical Analysis*, 2:205–223, 1965.

[73] Gutmann, J.S., Burgard, W., Fox, D., Konolige, K., "An Experimental Comparison of Localization Methods," *in Proceedings of the 1998 IEEE/RSJ International. Conference of Intelligent Robots and Systems* (IROS'98), Victoria, B.C., Canada, October 1998.

[74] Guttman, J.S., Konolige, K., "Incremental Mapping of Large Cyclic Environments," *in Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, Monterey, November 1999.

[75] Hashimoto, S., "Humanoid Robots in Waseda University - Hadaly-2 and WABIAN," *in IARP First International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, October 1998.

[76] Heale, A., Kleeman, L., "A Real Time DSP Sonar Echo Processor," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, Takamatsu, Japan, October 31–November 5, 2000.

[77] Horn, B.K.P., Schunck, B.G., "Determining Optical Flow," *Artificial Intelligence*, 17:185–203, 1981.

[78] Horswill, I., "Visual Collision Avoidance by Segmentation," *in Proceedings of IEEE International Conference on Robotics and Automation*, 902–909, 1995, IEEE Press, Munich, November 1994.

[79] Jacobs, R. and Canny, J., "Planning Smooth Paths for Mobile Robots," *in Proceeding. of the IEEE Conference on Robotics and Automation*, IEEE Press,1989, pp. 2–7.

[80] Jennings, J., Kirkwood-Watts, C., Tanis, C., "Distributed Map-making and Navigation in Dynamic Environments," *in Proceedings of the 1998 IEEE/RSJ Intl. Conference of Intelligent Robots and Systems (IROS'98)*, Victoria, B.C., Canada, October 1998.

[81] Jensfelt, P., Austin, D., Wijk, O., Andersson, M., "Feature Based Condensation for Mobile Robot Localization," *in Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, May 24–28, 2000.

[82] Kamon, I., Rivlin, E., Rimon, E., "A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, April 1996.

[83] Kelly, A., "Pose Determination and Tracking in Image Mosaic Based Vehicle Position Estimation," *in Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, Takamatsu, Japan, October 31–November 5, 2000.

[134] Simhon, S., Dudek, G., "A Global Topological Map Formed by Local Metric Maps," *in Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, B.C., Canada, October 1998.

[135] Simmons, R., "The Curvature Velocity Method for Local Obstacle Avoidance," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, April 1996.

[136] Smith, R., Self, M., Cheeseman, P., "Estimating Uncertain Spatial Relationships in Robotics," *Autonomous Robot Vehicles,* I. J. Cox and G. T. Wilfong (editors), Springer-Verlag, 1990, pp. 167–193.

[137] Sordalen, O.J., Canudas de Wi,t C., "Exponential Control Law for a Mobile Robot: Extension to Path Following," *IEEE Transactions on Robotics and Automation*, 9:837–842, 1993.

[138] Steinmetz, B.M., Arbter, K., Brunner, B., Landzettel, K., "Autonomous Vision-Based Navigation of the Nanokhod Rover," *in Proceedings of i-SAIRAS 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Montrea1, June 18–22, 2001.

[139] Stentz, A., "The Focussed D* Algorithm for Real-Time Replanning," in *Proceedings of IJCAI-95*, August 1995.

[140] Stevens, B.S., Clavel, R., Rey, L., "The DELTA Parallel Structured Robot, Yet More Performant through Direct Drive," *in Proceedings of the 23rd International Symposium on Industrial Robots*, Barcelona, October 1992, pp. 485–493.

[141] Takeda, H., Facchinetti, C., Latombe, J.C., "Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1002–1017, 1994.

[142] Thrun, S., Burgard, W., Fox, D., "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots." *Autonomous Robots*, 31:1–25. 1998.

[143] Thrun, S., et al., "Monrovia: A Second Generation Museum Tour-Guide Robot," *in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99)*, Detroit, May 1999.

[144] Thrun, S., Fox, D., Burgard, W. Dellaert, F., "Robust Monte Carlo Localization for Mobile Robots," *Artificial Intelligence*, 128:99–141, 2001.

[145] Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., Kuipers, B., "Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach," *in Proceedings of the National Conference on Artificial Intelligence (AAAI)*,1998.

[146] Tomasi, C., Shi, J., "Image Deformations Are Better Than Optical Flow." *Mathematical and Computer Modelling*, 24:165–175, 1996.

[147] Tomatis, N., Nourbakhsh, I., Siegwart, R., "Hybrid Simultaneous Localization and Map Building: A Natural Integration of Topological and Metric." *Robotics and Autonomous Systems* 44, 3–14, 2003.

[148] Tzafestas, C.S., Tzafestas, S.G., "Recent Algorithms for Fuzzy and Neurofuzzy Path Planning and Navigation of Autonomous Mobile Robots," *Systems-Science*, 25:25–39, 1999.

[149] Ulrich, I., Borenstein, J., "VFH*: Local Obstacle Avoidance with Look-Ahead Verification," *in Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, May 24–28, 2000.

Index

A

accuracy 1
action update 213
Aibo 27, 28
aliasing 182, 184
Asimo (Honda) 25
ASL approach 285-286, 290
autofocus 125

B

Bayes 214-215, 224, 233
behavior 27, 47-48, 60, 64, 92-93, 108, 119-120, 147, 153, 169, 191-194, 210, 223, 244, 249, 253, 258, 271, 276-277, 281, 292-303
belief
  multiple-hypothesis 194, 196-199, 212, 217, 222-223
  representation 11, 194-195, 214, 223
  single 194, 198
  single-hypothesis 194-196, 199
  state 198-227, 233-238, 258-259
  unique 196
biped 13, 15, 19-20, 24-26
blanking time 105
blooming 120, 183
blur circle 124, 127-128
bow leg 22, 23
bubble band 278-280, 288
Bug 272-276, 287

C

calibration 94, 131-132, 186, 190
camera
  axis 247
  geometry 123, 127-128
  optics 120, 123-125, 246
  parameter 119-120, 123
catadioptric camera 176-177
CCD camera 11, 90-92, 113, 117-122, 138, 164, 176, 183, 246 -247
center of mass (COM) 34, 36, 44, 143
chassis 32 - 80, 249, 275, 284, 291
Chips 209, 301
closed-world assumption 200, 202, 206
CMOS camera 11, 91-92, 113, 117-122, 138, 143-144, 178
CMUcam 143-145
CMVision 143-145
Cognachrome 143-145
computation 2, 8, 10, 17, 164, 181-185, 193, 257, 298
color sensing 142
completeness 227, 259, 261, 264-267, 270
configuration space 74, 206, 212, 214, 259-263, 267, 270, 282, 285-286
conjugate pair 131
constraint
  rolling 54, 57-63
  sliding 55, 57, 60, 63, 67, 71, 75-76
controllability 33-38
convolution 133-134, 165, 167
correlation 113, 136, 164, 188, 248-253
correspondence problem 115, 133-136, 142
cross-sensitivity 94-95, 98, 107, 120-121
curvature velocity 280, 289
cycle frequency 284
Cye 34-37, 206, 266, 302

D

dead reckoning 39-40, 43, 70, 98, 185, 245-246
decomposition 203-204, 217, 261, 264-265, 274, 292, 298
  approximate 206, 266
  control 295

hexapod 27
histogram 174- 177
angle 162, 246
color, image 177, 179, 246
polar 277-278
range 161
vector field 276
holonomic 40, 75-77, 260, 278, 286, 290
Honda Asimo 25
hopping
bow leg 22
Raibert 22
Ringrose 24
single leg 21
Hough transform 171-172, 178
hypothesis
multiple 194-202, 212, 217, 222
single 194, 196, 199, 202

I

image buffer chip 122
image fingerprint 178
image histogram 179, 246
image histogram 177
instantaneous center of rotation (ICR) 68
interface 126
inverse filtering 28
inverse kinematics 45
iris 119, 123-124

J

Jacobian 85, 150, 159, 189, 236-237, 241

K

Kalman filter 11, 198-199, 212-214, 227, 228--252
kernel 134-135, 165, 169
kinematic 17, 31, 47-48, 53-54, 58, 64, 69, 73-74, 83, 213, 224, 284, 287
analysis 80
constraints 48, 53, 61-62, 67-77, 280-285
control 81
forward 48, 51, 63
inverse 45
limitations 278
motion 67, 259

L

lane curvature 281, 289
Laplacian 133-135, 141, 165-167
laser rangefinder 91, 94, 104, 108, 110, 112, 122, 147, 153, 173, 223, 225, 229, 238-239, 264, 291
least-square 154-157, 229
lidar (light detection and ranging) 108
line extraction 154-155, 159, 202, 239
linearity 93, 120
localization 2, 8, 10, 97, 101, 181-184, 191-194, 198, 201, 207-214, 245-246, 291
beacon 248
control 291
global 246-247
Kalman filter 11, 212-214, 227-229, 233-234, 238, 244
landmark-based 245
Markov 11, 212-218, 227-227
mosaic-based 247
probabilistic 212
representation 191
route-based 249
sensor 103-104, 142, 162, 166, 177, 211, 264
localization and mapping (SLAM) 250, 300
locomotion 13-17
biological 14
biped 15
legged 17, 19, 21
mechanism 14, 45, 211
specific power 15
wheeled 30-31, 42

M

Mahalanobis 236, 242
maneuverability 30, 33, 37-43, 48, 63, 67, 72, 74, 77
manipulator 1, 47, 259
Markov assumption 216
Markov localization 212-217, 223-227, 233
mean square 147
minimum energy algorithm 180
mobility 2
motion control 80-82, 181, 191, 298
motion field 138, 139
motorization 80

N

Nanokhod 42