5 Mobile Robot Localization

5.1 Introduction

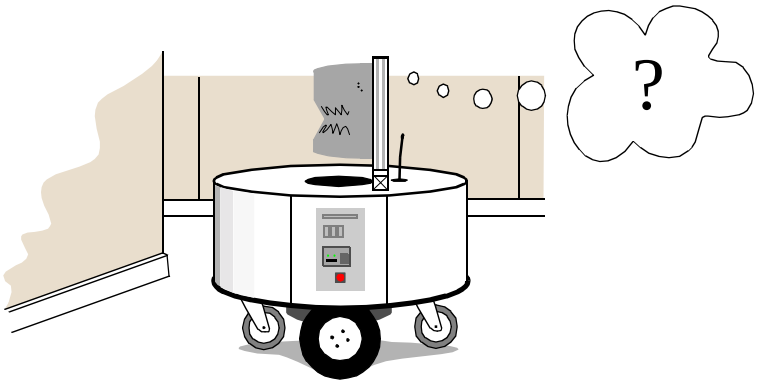


Fig 5.1 *Where am I?*

Navigation is one of the most challenging competencies required of a mobile robot. Success in navigation requires success at the four building blocks of navigation (fig. 5.2): *perception*- the robot must interpret its sensors to extract meaningful data; *localization*- the robot must determine its position in the environment; *cognition*- the robot must decide how to act to achieve its goals; and *motion control*- the robot must modulate its motor outputs to achieve the desired trajectory.

Of these four components, localization has received the greatest research attention in the past decade and, as a result, significant advances have been made on this front. In this chapter, we will explore the successful localization methodologies of recent years. First, Section 5.2 describes how sensor and effector uncertainty is responsible for the difficulties of localization. Then, Section 5.3 describes two extreme approaches to dealing with the challenge of robot localization: avoiding localization altogether, and performing explicit map-based localization. The remainder of the chapter discusses the question of representation, then presents case studies of successful localization systems using a variety of representations and techniques to achieve mobile robot localization competence.

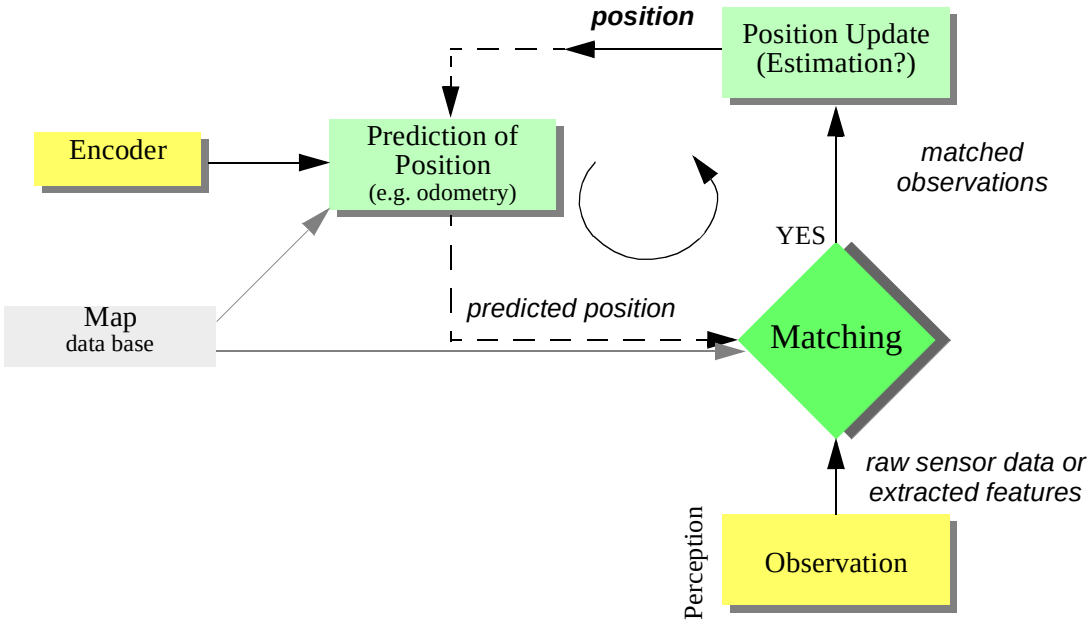


Fig 5.2 *General schematic for mobile robot localization.*

5.2 The Challenge of Localization: noise and aliasing

If one could attach an accurate *GPS* (Global Position System) sensor to a mobile robot, much of the localization problem would be obviated. The *GPS* would inform the robot of its exact position and orientation, indoors and outdoors, so that the answer to the question, "Where am I?" would always be immediately available. Unfortunately, such a sensor is not currently practical. The existing *GPS* network provides accuracy to within several meters, which is unacceptable for localizing human-scale mobile robots as well as miniature mobile robots such as desk robots and the body-navigating nano-robots of the future. Furthermore, *GPS* technologies cannot function indoors or in obstructed areas and are thus limited in their workspace.

But, looking beyond the limitations of *GPS*, localization implies more than knowing one's absolute position in the Earth's reference frame. Consider a robot that is interacting with humans. This robot may need to identify its absolute position, but its relative position with respect to target humans is equally important. Its localization task can include identifying humans using its sensor array, then computing its relative position to the humans. Furthermore, during the *Cognition* step a robot will select a strategy for achieving its goals. If it intends to reach a particular location, then localization may not be enough. The robot may need to acquire or build an environmental model, a *map*, that aids it in planning a path to the goal. Once again, localization means more than simply determining an absolute pose in space; it means building a map, then identifying the robot's position relative to that map.

Clearly, the robot's sensors and effectors play an integral role in all the above forms of localization. It is because of the inaccuracy and incompleteness of these sensors and effectors that localization poses difficult challenges. This section identifies important aspects of this sensor and effector suboptimality.

5.2.1 Sensor Noise

Sensors are the fundamental robot input for the process of *perception*, and therefore the degree to which sensors can discriminate world state is critical. *Sensor noise* induces a limitation on the consistency of sensor readings in the same environmental state and, therefore, on the number of useful bits available from each sensor reading. Often, the source of sensor noise problems is that some environmental features are not captured by the robot's representation and are thus overlooked.

For example, a vision system used for indoor navigation in an office building may use the color values detected by its color CCD camera. When the sun is hidden by clouds, the illumination of the building's interior changes due to windows throughout the building. As a result, hue values are not constant. The color CCD appears noisy from the robot's perspective as if subject to random error, and the hue values obtained from the CCD camera will be unusable, unless the robot is able to note the position of the Sun and clouds in its representation.

Illumination dependency is only one example of the apparent noise in a vision-based sensor system. Picture jitter, signal gain, blooming and blurring are all additional sources of noise, potentially reducing the useful content of a color video image.

Consider the noise level (i.e. apparent random error) of ultrasonic range-measuring sensors (e.g. sonars) as we discussed in Section 4.1.2.3. When a sonar transducer emits sound towards a relatively smooth and angled surface, much of the signal will coherently reflect away, failing to generate a return echo. Depending on the material characteristics, a small amount of energy may return nonetheless. When this level is close to the gain threshold of the sonar sensor, then the sonar will, at times, succeed and, at other times, fail to detect the object. From the robot's perspective, a virtually unchanged environmental state will result in two different possible sonar readings: one short, and one long.

The poor signal to noise ratio of a sonar sensor is further confounded by interference between multiple sonar emitters. Often, research robots have between 12 to 48 sonars on a single platform. In acoustically reflective environments, multipath interference is possible between the sonar emissions of one tranducer and the echo detection circuitry of another transducer. The result can be dramatically large errors (i.e. underestimation) in ranging values due to a set of coincidental angles. Such errors occur rarely, less than 1% of the time, and are virtually random from the robot's perspective.

In conclusion, sensor noise reduces the useful information content of sensor readings. Clearly, the solution is to take multiple readings into account, employing temporal fusion or multi-sensor fusion to increase the overall information content of the robot's inputs.

5.2.2 Sensor Aliasing

A second shortcoming of mobile robot sensors causes them to yield little information content, further exacerbating the problem of perception and, thus, localization. The problem, known as *sensor aliasing*, is a phenomenon that humans rarely encounter. The human sensory system, particularly the visual system, tends to receive unique inputs in each unique local state. In other words, every different place looks different. The power of this unique mapping is only apparent when one considers situations where this fails to hold. Consider moving through an unfamiliar building that is completely dark. When the visual system sees only black, one's localization system quickly degrades. Another useful example is that of a human-sized maze made from tall hedges. Such mazes have been created for centuries, and humans find them extremely difficult to solve without landmarks or clues because, without visual uniqueness, human localization competence degrades rapidly.

In robots, the non-uniqueness of sensors readings, or *sensor aliasing*, is the norm and not the exception. Consider a narrow-beam rangefinder such as ultrasonic or infrared rangefinders. This sensor provides range information in a single direction without any additional data regarding material composition such as color, texture and hardness. Even for a robot with several such sensors in an array, there are a variety of environmental states that would trigger the same sensor values across the array. Formally, there is a many-to-one mapping from environmental states to the robot's perceptual inputs. Thus, the robot's percepts cannot distinguish from among these many states. A classical problem with sonar-based robots involves distinguishing between humans and inanimate objects in an indoor setting. When facing an apparent obstacle in front of itself, should the robot say "Excuse me" because the obstacle may be a moving human, or should the robot plan a path around the object because it may be a cardboard box? With sonar alone, these states are aliased and differentiation is impos-

sible.

The problem posed to navigation because of sensor aliasing is that, even with noise-free sensors, the amount of information is generally insufficient to identify the robot's position from a single percept reading. Thus techniques must be employed by the robot programmer that base the robot's localization on a series of readings and, thus, sufficient information to recover the robot's position over time.

5.2.3 Effector Noise

The challenges of localization do not lie with sensor technologies alone. Just as robot sensors are noisy, limiting the information content of the signal, so robot effectors are also noisy. In particular, a single action taken by a mobile robot may have several different possible results, even though from the robot's point of view the initial state before the action was taken is well-known.

In short, mobile robot effectors introduce uncertainty about future state. Therefore the simple act of moving tends to increase the uncertainty of a mobile robot. There are, of course, exceptions. Using *cognition*, the motion can be carefully planned so as to minimize this effect, and indeed sometimes to actually result in more certainty. Furthermore, when the robot actions are taken in concert with careful interpretation of sensory feedback, it can compensate for the uncertainty introduced by noisy actions using the information provided by the sensors.

First, however, it is important to understand the precise nature of the effector noise that impacts mobile robots. It is important to note that, from the robot's point of view, this error in motion is viewed as error in odometry, or the robot's inability to estimate its own position over time using knowledge of its kinematics and dynamics. The true source of error generally lies in an incomplete model of the environment. For instance, the robot does not model the fact that the floor may be sloped, the wheels may slip, and a human may push the robot. All of these un-modeled sources of error result in inaccuracy between the physical motion of the robot, the intended motion of the robot and the proprioceptive sensor estimates of motion.

In odometry (wheel sensors only) and dead reckoning (also heading sensors) the position update is based on *proprioceptive* sensors. The movement of the robot, sensed with wheel encoders and /or heading sensors is integrated to compute position. Because the sensor measurement errors are integrated, the position error accumulates over time. Thus the position has to be updated from time to time by other localization mechanisms. Otherwise the robot is not able to maintain a meaningful position estimate in long run.

In the following we will concentrate on odometry based on the wheel sensor readings of a differential drive robot only (see also [3, 40, 41]). Using additional heading sensors (e.g. gyroscope) can help to reduce the cumulative errors, but the main problems remain the same.

There are many sources of odometric error, from environmental factors to resolution:

- Limited resolution during integration (time increments, measurement resolution, etc.)
- Misalignment of the wheels (deterministic)

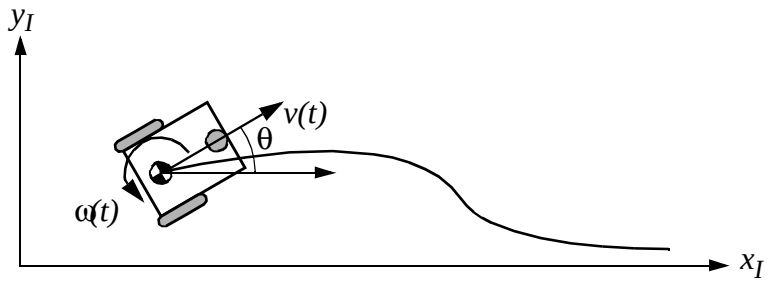


Fig 5.3 Movement of a differential drive robot

- Unequal wheel diameter (deterministic)
- Variation in the contact point of the wheel
- Unequal floor contact (slipping, non-planar 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 ***non-deterministic*** *(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:

- Range error: integrated path length (distance) of the robots movement
  -> sum of the wheel movements
- Turn error: similar to range error, but for turns
  -> difference of the wheel motions
- 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 contribute 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:

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

$$\Delta y = \Delta s \sin(\theta + \Delta\theta / 2) \tag{5.3}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{5.5}$$

where:

$(\Delta x;\Delta y;\Delta\theta)$: Path traveled in the last sampling interval

$\Delta s_r;\Delta s_l$: Traveled distances for right and left wheel respectively

b: Distance between the two wheels of differential drive robot

Thus we get the updated position p':

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta / 2) \\ \Delta s \sin(\theta + \Delta\theta / 2) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta / 2) \\ \Delta s \sin(\theta + \Delta\theta / 2) \\ \Delta\theta \end{bmatrix} \tag{5.6}$$

By using the relation for $(\Delta s;\Delta\theta)$ of equations (5.4) and (5.5) we further obtain the basic equation for odometric position update (for differential drive robots):

$$p' = f(x,\ y,\ \theta,\ \Delta s_r,\ \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.7}$$

As we discussed earlier, odometric position updates can give only a very rough estimate of the actual position. Due to integration errors of the uncertainties of p and the motion errors during the incremental motion $(\Delta s_r;\Delta s_l)$ the position error based on odometry integration grows with time.

In the next step we will establish an error model for the integrated position p' to obtain the covariance matrix $\Sigma_{p'}$ of the odometric position estimate. To do so, we assume that at the

starting point the initial covariance matrix Σ_p is known. For the motion increment $(\Delta s_r;\Delta s_l)$ we assume the following covariance matrix Σ_Δ:

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.8}$$

where Δs_r and Δs_l are the distances travelled by each wheel, and k_r, k_l are error constants representing the non-deterministic parameters of the motor drive and the wheel-floor interaction. As you can see in equation (5.8) we made the following assumption:

- The two errors of the individually driven wheels are independent[1]
- The errors are proportional to the absolute value of the traveled distances $(\Delta s_r;\Delta s_l)$.

These assumptions, while not perfect, are suitable and will thus be used for the further development of the error model. The *motion errors* are due to unprecise movement because of deformation of wheel, slippage, unequal floor, errors in encoders, *et cetera*. The values for the error constants k_r and k_l depend on the robot and the environment and should be experimentally established by performing and analyzing representative movements.

If we assume that p and $\Delta_{rl} = (\Delta s_r;\Delta s_l)$ are uncorrelated and the derivation of f (equ. (5.7)) is reasonably approximated by the first order Taylor expansion (linearization) we conclude, using the error propagation law (see section 4.2.3):

$$\Sigma_{p'} = \nabla_p f \cdot \Sigma_p \cdot \nabla_p f^T + \nabla_{\Delta_{rl}} f \cdot \Sigma_\Delta \cdot \nabla_{\Delta_{rl}} f^T \tag{5.9}$$

The covariance matrix Σ_p is, of course, always given by the $\Sigma_{p'}$ of the previous step, and can thus be calculated after specifying an initial value (e.g. 0).

Using equation (5.7) we can develop the two *Jacobians* $F_p = \nabla_p f$ and $F_{\Delta_{rl}} = \nabla_{\Delta_{rl}} f$:

$$F_p = \nabla_p f = \nabla_p (f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta / 2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta / 2) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

1. If there is more knowledge regarding the actual robot kinematics, the correlation terms of the covariance matrix could also be used.

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos(\theta + \Delta\theta/2) - \frac{\Delta s}{2b}\sin(\theta + \Delta\theta/2) & \frac{1}{2}\cos(\theta + \Delta\theta/2) + \frac{\Delta s}{2b}\sin(\theta + \Delta\theta/2) \\ \frac{1}{2}\sin(\theta + \Delta\theta/2) + \frac{\Delta s}{2b}\cos(\theta + \Delta\theta/2) & \frac{1}{2}\sin(\theta + \Delta\theta/2) - \frac{\Delta s}{2b}\cos(\theta + \Delta\theta/2) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \quad (5.11)$$

The details for arriving at equation (5.11) are:

$$F_{\Delta_{rl}} = \nabla_{\Delta_{rl}} f = \begin{bmatrix} \frac{\partial f}{\partial \Delta s_r} & \frac{\partial f}{\partial \Delta s_l} \end{bmatrix} = \ldots \quad (5.12)$$

$$\ldots = \begin{bmatrix} \frac{\partial \Delta s}{\partial \Delta s_r}\cos(\theta + \Delta\theta/2) + (-\sin(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta s}{\partial \Delta s_l}\cos(\theta + \Delta\theta/2) + (-\sin(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_l} \\ \frac{\partial \Delta s}{\partial \Delta s_r}\sin(\theta + \Delta\theta/2) + (\cos(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta s}{\partial \Delta s_l}\sin(\theta + \Delta\theta/2) + (\cos(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_l} \\ \frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta\theta}{\partial \Delta s_l} \end{bmatrix} \quad (5.13)$$

and with

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \quad ; \quad \Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \quad (5.14)$$

$$\frac{\partial \Delta s}{\partial \Delta s_r} = \frac{1}{2} \quad ; \quad \frac{\partial \Delta s}{\partial \Delta s_l} = \frac{1}{2} \quad ; \quad \frac{\partial \Delta\theta}{\partial \Delta s_r} = \frac{1}{b} \quad ; \quad \frac{\partial \Delta\theta}{\partial \Delta s_l} = -\frac{1}{b} \quad (5.15)$$

we obtain equation (5.11).

Figures 5.4 and 5.5 show typical examples of how the position errors grow with time. The results have been computed using the error model presented above.

Once the error model has been established, the error parameters must be specified. One can compensate for deterministic errors properly calibrating the robot. However the error parameters specifying the non-deterministic errors can only be quantified by statistical (repetitive) measurements. A detailed discussion of odometric errors and a method for calibration and quantification of deterministic and non-deterministic errors can be found in [4].

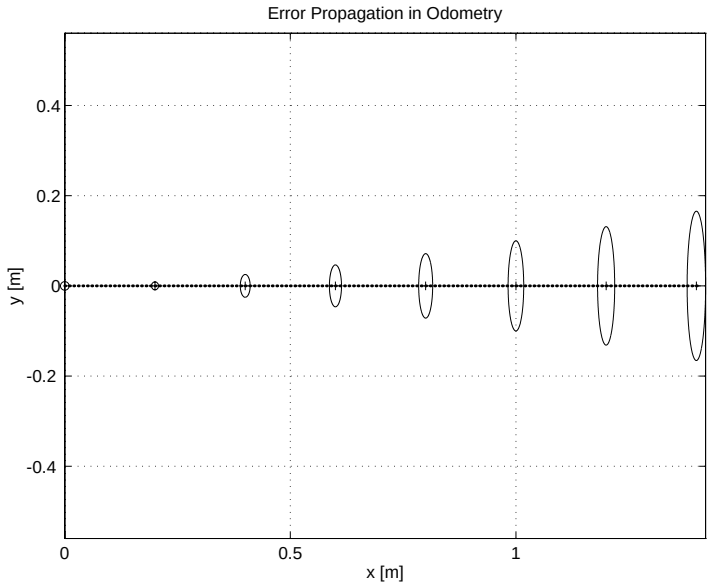


Fig 5.4 *Growth of the pose uncertainty for straight line movement: Note that the uncertainty in y grows much faster than in the direction of movement. This results from the integration of the uncertainty about the robot's orientation. The ellipses drawn around the robot positions represent the uncertainties in the x,y direction (e.g. 3σ). The uncertainty of the orientation θ is not represented in the picture although its effect can be indirectly observed.*

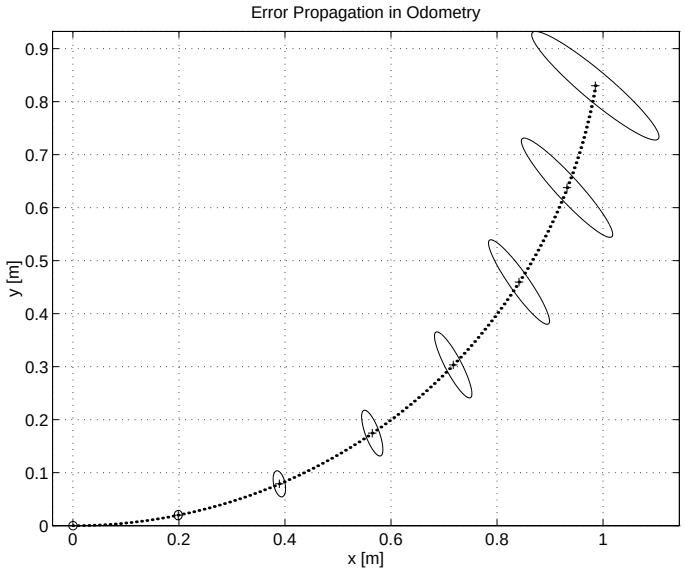


Fig 5.5 *Growth of the pose uncertainty for circular movement (r=const): Again, the uncertainty perpendicular to the movement grows much faster than that in the direction of movement. Note that the main axis of the uncertainty ellipsis does not remain perpendicular to the direction of movement.*

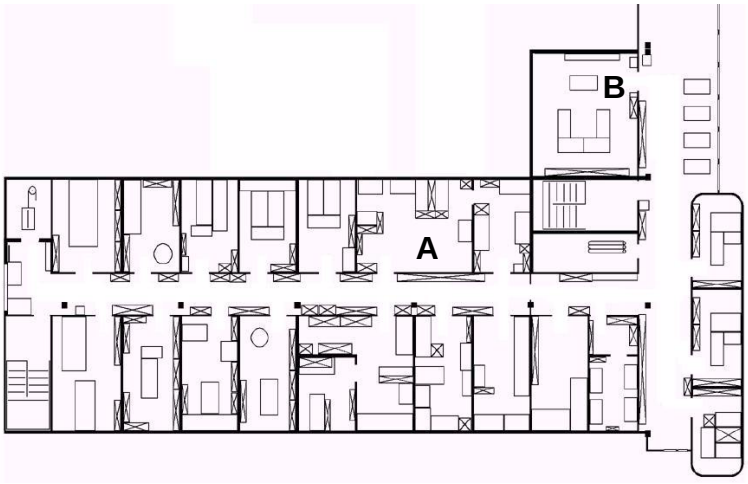


Fig 5.6 A Sample Environment

5.3 To Localize or Not to Localize: localization-based navigation versus programmed solutions

Figure 5.6 depicts a standard indoor environment that a mobile robot navigates. Suppose that the mobile robot in question must deliver messages between two specific rooms in this environment: rooms ***A*** and ***B***. In creating a navigation system, it is clear that the mobile robot will need sensors and a motion control system. Sensors are absolutely required to avoid hitting moving obstacles such as humans, and some motion control system is required so that the robot can deliberately move.

It is less evident, however, whether or not this mobile robot will require a *localization system*. Localization may seem mandatory in order to successfully navigate between the two rooms. It is through localizing on a map, after all, that the robot can hope to recover its position and detect when it has arrived at the goal location. It is true that, at the least, the robot must have a way of detecting the goal location. However, explicit localization with reference to a map is not the only strategy that qualifies as a goal detector.

An alternative, espoused by the behavior-based community, suggests that, since sensors and effectors are noisy and information-limited, one should avoid creating a geometric map for localization. Instead, this community suggests designing sets of behaviors that together result in the desired robot motion. Fundamentally, this approach avoids explicit reasoning about localization and position, and thus generally avoids explicit path planning as well.

This technique is based on a belief that there exists a procedural solution to the particular navigation problem at hand. For example, in Fig. 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 disadvan-

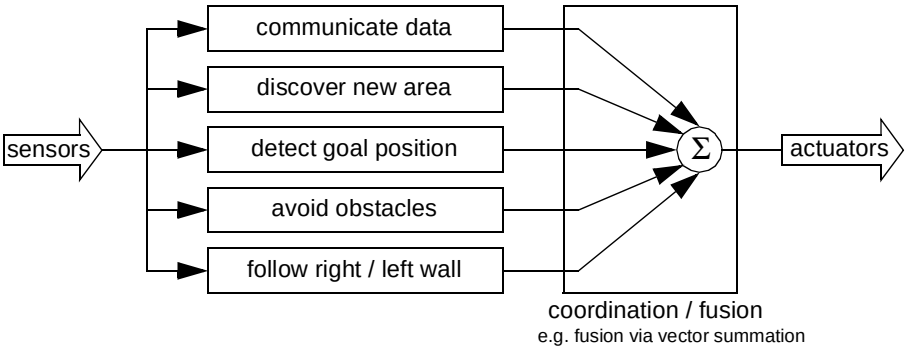


Fig 5.7 An Architecture for Behavior-based Navigation

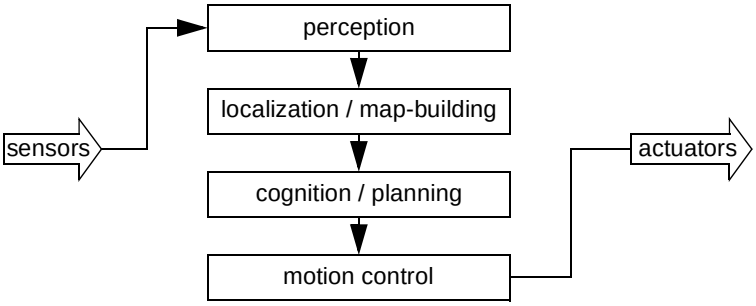


Fig 5.8 An Architecture for Map-based (or model-based) Navigation

tages, 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 re-tune all of the existing behaviors again to ensure that the new interactions with the freshly introduced behavior are all stable.

In contrast to the behavior-based approach, the map-based approach includes both *localization* and *cognition* modules (see Fig. 5.8). In map-based navigation, the robot explicitly attempts to localize by collecting sensor data, then updating some belief about its position with respect to a map of the environment. The key advantages of the map-based approach for navigation are as follows:

- The explicit, map-based concept of position makes the system's belief about position transparently available to the human operators.
- The existence of the map itself represents a medium for communication between human and robot: the human can simply give the robot a new map if the robot goes to

a new environment.

- 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 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, or 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 expressed as a single unique point on the map. In Fig. 5.10, three examples of a single hypothesis belief are shown using three different map representations of the same actual environment (fig. 5.10a). In 5.10b, a single point is geometrically annotated as the robot's position in a continuous two-dimensional geometric map. In 5.10c, the map is a discrete, tessellated map, and the position is noted at the same level of fidelity as the map cell size. In 5.10d, the map is not geometrical at all but abstract and topological. In this case, the single hypothesis of position involves identifying a single node *i* in the topological graph as the robot's position.

The principal advantage of the single hypothesis representation of position stems from the fact that, given a unique belief, there is no position ambiguity. The unambiguous nature of this representation facilitates decision-making at the robot's cognitive level (e.g. path planning). The robot can simply assume that its belief is correct, and can then select its future actions based on its unique position.

Just as decision-making is facilitated by a single-position hypothesis, so updating the robot's belief regarding position is also facilitated, since the single position must be updated by definition to a new, single position. The challenge with this position update approach, which ultimately is the principal disadvantage of single-hypothesis representation, is that robot motion often induces uncertainty due to effectory and sensory noise. Therefore, forcing the position update process to always generate a *single* hypothesis of position is challenging

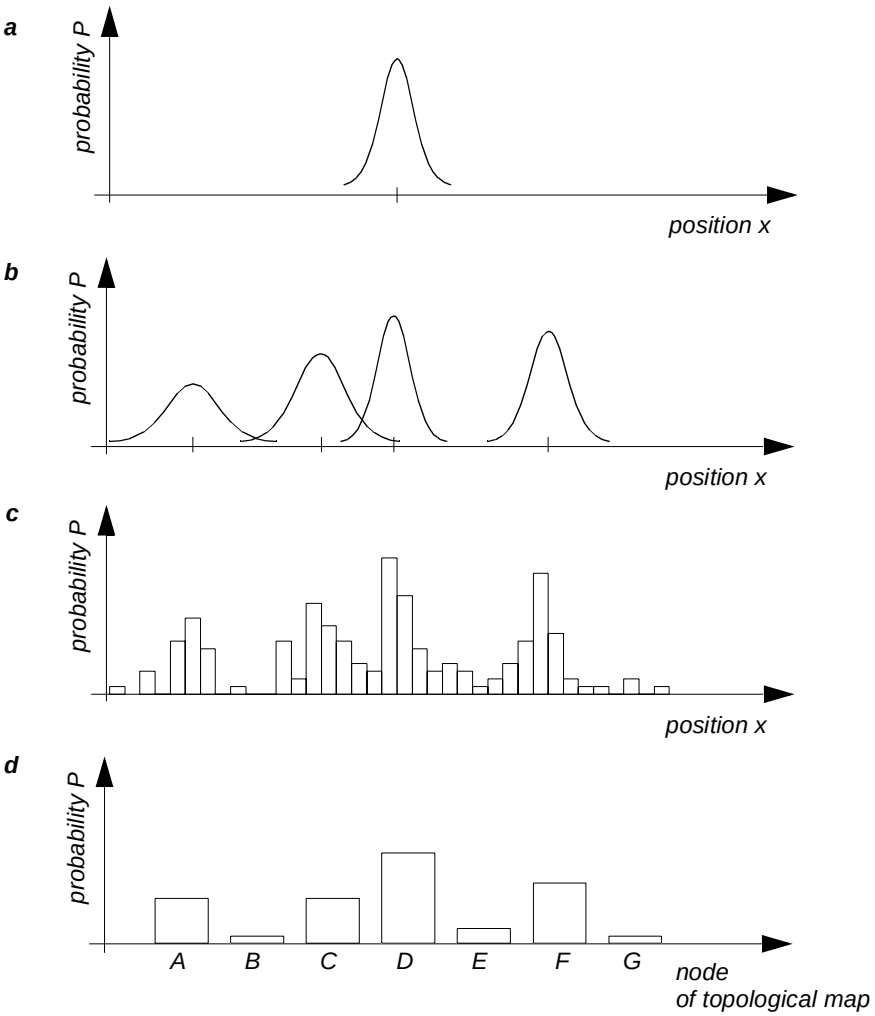


Fig 5.9 *Belief representation regarding the robot position (1 dimensional) in continuous and discretized (tessellated) maps.*
a)Continuous map with multiple hypothesis belief, e.g. single Gaussian centered at a single continuous value
b)Continuous map with multiple hypothesis belief, e.g. multiple Gaussians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

and, often, impossible.

5.4.2 Multiple Hypothesis Belief

In the case of multiple hypothesis beliefs regarding position, the robot tracks not just a single possible position but a possibly infinite set of positions.

In one simple example originating in the work of Jean-Claude Latombe [5, 89], the robot's position is described in terms of a convex polygon positioned in a two-dimensional map of the environment. This multiple hypothesis representation communicates the set of possible robot positions geometrically, with no preference ordering over the positions. Each point in

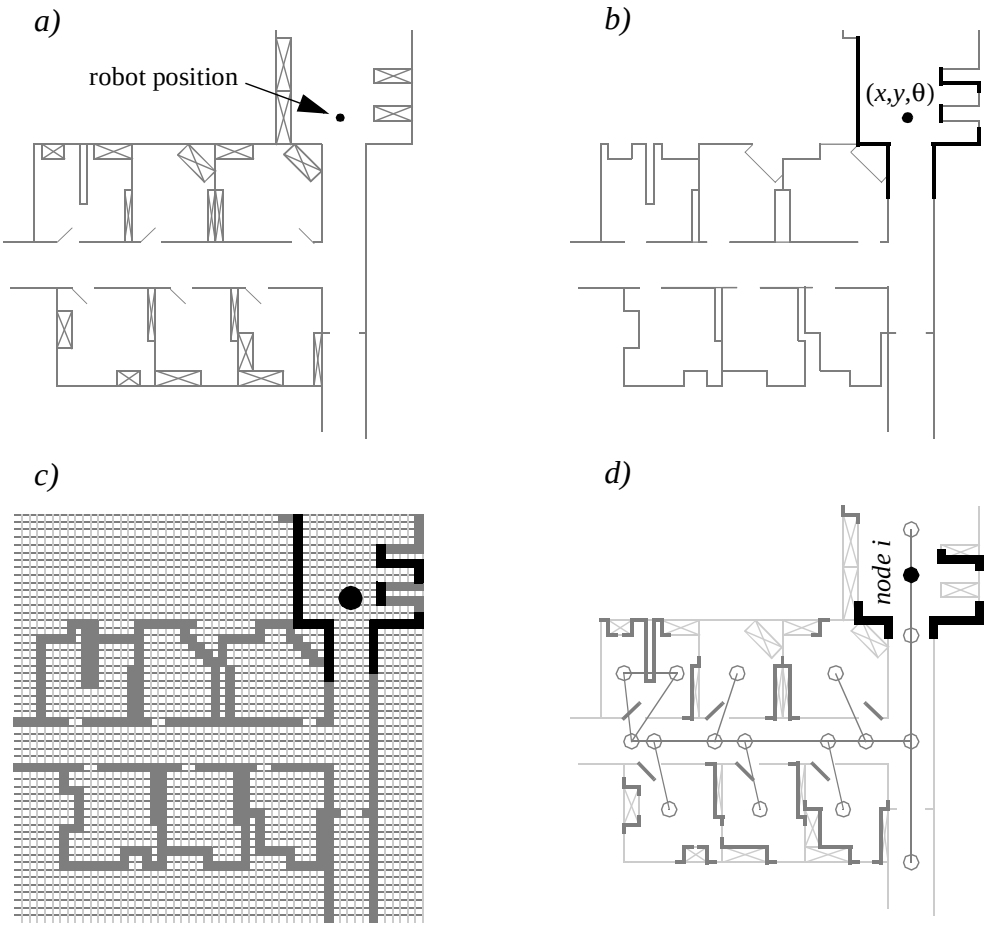
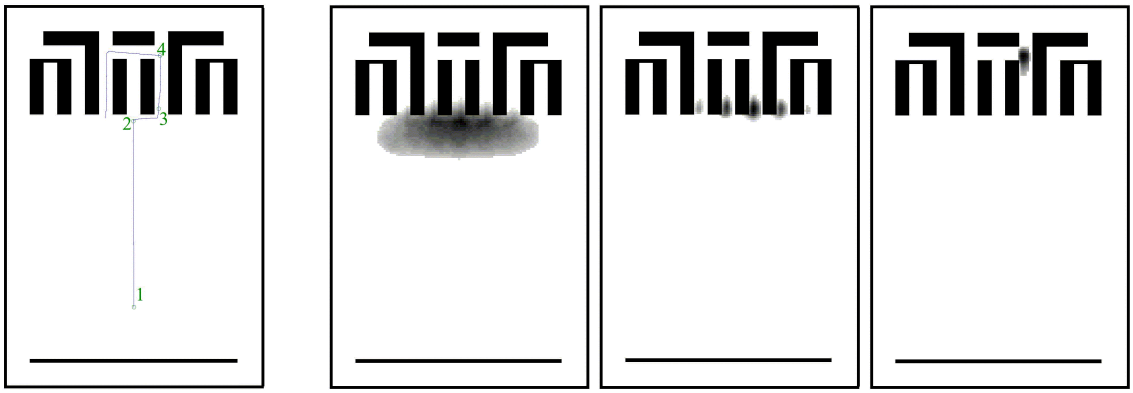


Fig 5.10 *Three examples of single hypotheses of position using different map representation.*
a) real map with walls, doors and furniture
b) line-based map
-> around 100 lines with two parameters
c) occupancy grid based map
-> around 3000 gird cells sizing 50x50 cm
d) topological map using line features (Z/S-lines) and doors
-> around 50 features and 18 nodes

the map is simply either contained by the polygon and, therefore, in the robot's belief set, or outside the polygon and thereby excluded. Mathematically, the position polygon serves to partition the space of possible robot positions. Such a polygonal representation of the multiple hypothesis belief can apply to a continuous, geometric map of the environment or, alternatively, to a tessellated, discrete approximation to the continuous environment.

It may be useful, however, to incorporate some ordering on the possible robot positions, capturing the fact that some robot positions are likelier than others. A strategy for representing a continuous multiple hypothesis belief state along with a preference ordering over possible positions is to model the belief as a mathematical distribution. For example, [42,47] notate the robot's position belief using an $\{X,Y\}$ point in the two-dimensional environment as the mean μ plus a standard deviation parameter σ, thereby defining a Gaussian distribution. The intended interpretation is that the distribution at each position represents the probability



Path of the robot *Belief states at positions 2, 3 and 4*

Fig 5.11 *Example of multiple hypothesis tracking (courtesy of W. Burgard [43]). The belief state that is largely distributed becomes very certain after moving to position 4*

assigned to the robot being at that location. This representation is particularly amenable to mathematically defined tracking functions, such as the Kalman Filter, that are designed to operate efficiently on Gaussian distributions.

An alternative is to represent the set of possible robot positions, not using a single Gaussian probability density function, but using discrete markers for each possible position. In this case, each possible robot position is individually noted along with a confidence or probability parameter (See Fig. (5.11)). In the case of a highly tessellated map this can result in thousands or even tens of thousands of possible robot positions in a single belief state.

The key advantage of the multiple hypothesis representation is that the robot can explicitly maintain uncertainty regarding its position. If the robot only acquires partial information regarding position from its sensors and effectors, that information can conceptually be incorporated in an updated belief.

A more subtle advantage of this approach revolves around the robot's ability to explicitly measure its own degree of uncertainty regarding position. This advantage is the key to a class of localization and navigation solutions in which the robot not only reasons about reaching a particular goal, but reasons about the future trajectory of its own belief state. For instance, a robot may choose paths that minimize its future position uncertainty. An example of this approach is [90], in which the robot plans a path from point *A* to *B* that takes it near a series of landmarks in order to mitigate localization difficulties. This type of explicit reasoning about the effect that trajectories will have on the quality of localization requires a multiple hypothesis representation.

One of the fundamental disadvantages of the multiple hypothesis approaches involves decision-making. If the robot represents its position as a region or set of possible positions, then how shall it decide what to do next? Figure 5.11 provides an example. At position 3, the robot's belief state is distributed among 5 hallways separately. If the goal of the robot is to travel down one particular hallway, then given this belief state what action should the robot choose?

The challenge occurs because some of the robot's possible positions imply a motion trajec-

tory that is inconsistent with some of its other possible positions. One approach that we will see in the case studies below is to assume, for decision-making purposes, that the robot is physically at the most probable location in its belief state, then to choose a path based on that current position. But this approach demands that each possible position have an associated probability.

In general, the right approach to such a decision-making problems would be to decide on trajectories that eliminate the ambiguity explicitly. But this leads us to the second major disadvantage of the multiple hypothesis approaches. In the most general case, they can be computationally very expensive. When one reasons in a three dimensional space of discrete possible positions, the number of possible belief states in the single hypothesis case is limited to the number of possible positions in the 3D world. Consider this number to be N. When one moves to an arbitrary multiple hypothesis representation, then the number of possible belief states is the power set of N, which is far larger: 2^N. Thus explicit reasoning about the possible trajectory of the belief state over time quickly becomes computationally untenable as the size of the environment grows.

There are, however, specific forms of multiple hypothesis representations that are somewhat more constrained, thereby avoiding the computational explosion while allowing a limited type of multiple hypothesis belief. For example, if one assumes a Gaussian distribution of probability centered at a single position, then the problem of representation and tracking of belief becomes equivalent to Kalman Filtering, a straightforward mathematical process described below. Alternatively, a highly tessellated map representation combined with a limit of 10 possible positions in the belief state, results in a discrete update cycle that is, at worst, only 10x more computationally expensive than single hypothesis belief update.

In conclusion, the most critical benefit of the multiple hypothesis belief state is the ability to maintain a sense of position while explicitly annotating the robot's uncertainty about its own position. This powerful representation has enabled robots with limited sensory information to navigate robustly in an array of environments, as we shall see in the case studies below.

5.5 Map Representation

The problem of representing the environment in which the robot moves is a dual of the problem of representing the robot's possible position or positions. Decisions made regarding the environmental representation can have impact on the choices available for robot position representation. Often the fidelity of the position representation is bounded by the fidelity of the map.

Three fundamental relationships must be understood when choosing a particular map representation:

- The precision of the map must appropriately match the precision with which the robot needs to achieve its goals.
- The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors.
- The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization and navigation.

In the following sections, we identify and discuss critical design choices in creating a map representation. Each such choice has great impact on the relationships listed above and on the resulting robot localization architecture. As we will see, the choice of possible map representations is broad. Selecting an appropriate representation requires understanding all of the trade-offs inherent in that choice as well as understanding the specific context in which a particular mobile robot implementation must perform localization. In general, the environment representation and model can be roughly classified as presented in chapter 4.3.

5.5.1 Continuous Representations

A continuous-valued map is one method for *exact* decomposition of the environment. The position of environmental features can be annoted precisely in continuous space. Mobile robot implementations to date use continuous maps only in two dimensional representations, as further dimensionality can result in computational explosion.

A common approach is to combine the exactness of a continuous representation with the compactness of the *closed world assumption*. This means that one assumes that the representation will specify all environmental objects in the map, and that any area in the map that is devoid of objects has no objects in the corresponding portion of the environment. Thus, the total storage needed in the map is proportional to the density of objects in the environment, and a sparse environment can be represented by a low-memory map.

One example of such a representation, shown in Figure 5.12, is a 2D representation in which polygons represent all obstacles in a continuos-valued coordinate space. This is similar to the method used by Latombe [5, 113] and others to represent environments for mobile robot path planning techniques.

In the case of [5, 113], most of the experiments are in fact simulations run exclusively within the computer's memory. Therefore, no real effort would have been expended to attempt to use sets of polygons to describe a real-world environment, such as a park or office building.

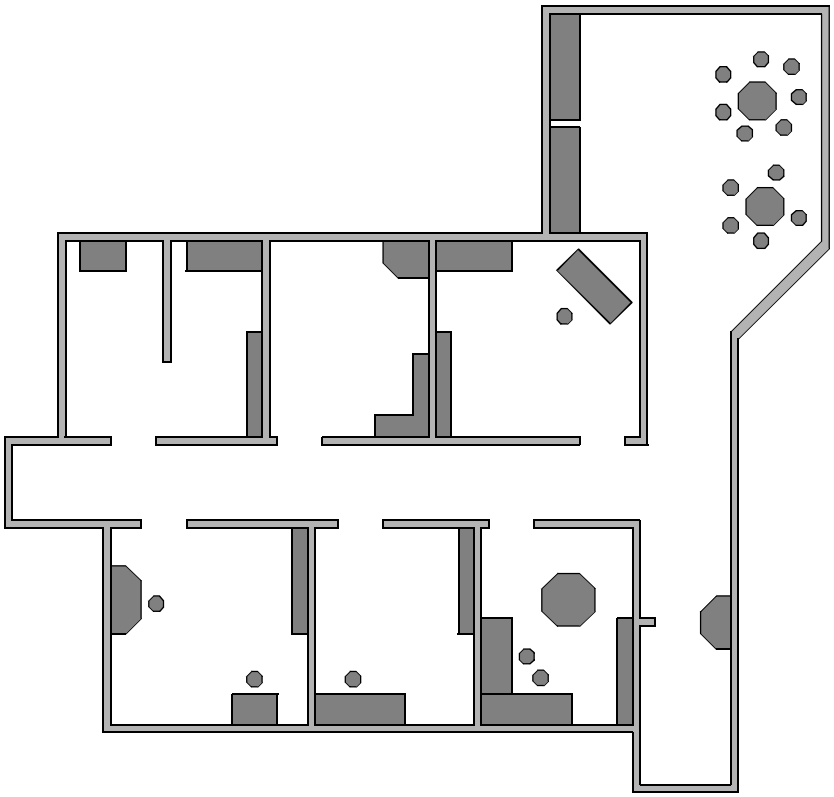


Fig 5.12 A continous representation using polygons as environmental obstacles

In other work in which real environments must be captured by the maps, one sees a trend toward selectivity and abstraction. The human map-maker tends to capture on the map, for localization purposes, only objects that can be detected by the robot's sensors and, furthermore, only a subset of the features of real-world objects.

It should be immediately apparent that geometric maps can capably represent the physical locations of objects without referring to their texture, color, elasticity, or any other such secondary features that do not related directly to position and space. In addition to this level of simplification, a mobile robot map can further reduce memory usage by capturing only aspects of object geometry that are immediately relevant to localization. For example all objects may be approximated using very simple convex polygons, sacrificing map felicity for the sake of computational speed.

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 strikes. 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 repre-

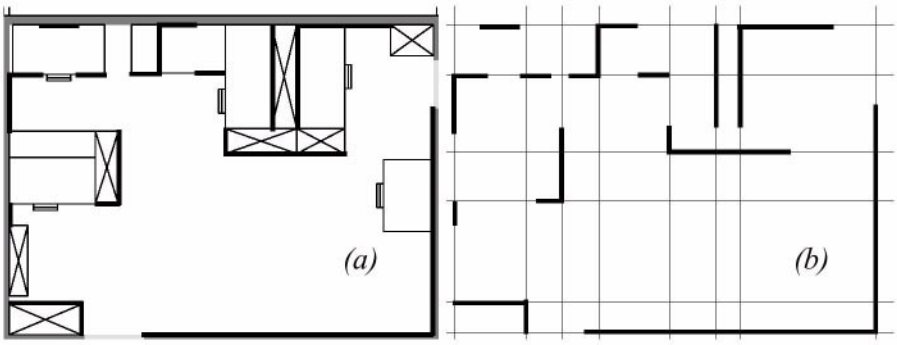


Fig 5.13 *Example of a continuous-valued line representation of EPFL.*
left: real map
right: representation with a set of infinite lines

sentation. 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 continuos 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 5.30, in which the position of the robot is depicted by an oval bounding area.

Yet, the continuous representation does not disallow representation of position in the form of a discrete set of possible positions. For instance, in [111] the robot position belief state is captured by sampling nine continuous-valued positions from within a region near the robot's best known position. This algorithm captures, within a continuous space, a discrete sampling of possible robot positions.

In summary, the key advantage of a continuous map representation is the potential for high accuracy and expressiveness with respect to the environmental configuration as well as the robot position within that environment. The danger of a continuous representation is that the map may be computationally costly. But this danger can be tempered by employing abstraction and capturing only the most relevant environmental features. Together with the use of the *closed world assumption*, these techniques can enable a continuous-valued map to be no more costly, and sometimes even less costly, than a standard discrete representation.

5.5.2 Decomposition Strategies

In the section above, we discussed one method of simplification, in which the continuous map representation contains a set of infinite lines that approximate real-world environmental lines based on a two-dimensional slice of the world. Basically this transformation from the real world to the map representation is a filter that removes all non-straight data and furthermore extends line segment data into infinite lines that require fewer parameters.

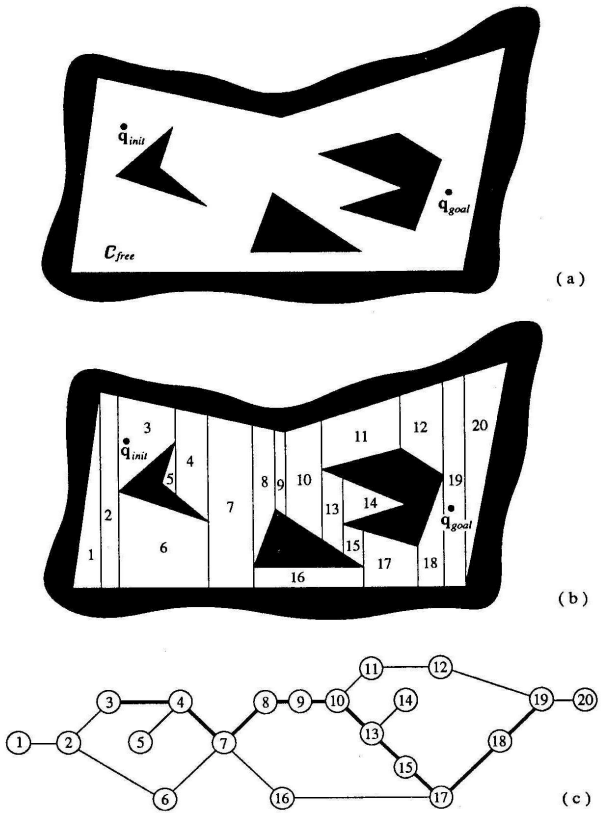


Fig 5.14 *Example of exact cell decomposition.*

A more dramatic form of simplification is *abstraction*: a general decomposition and selection of environmental features. In this section, we explore decomposition as applied in its more extreme forms to the question of map representation.

Why would one radically decompose the real environment during the design of a map representation? The immediate disadvantage of decomposition and abstraction is the loss of fidelity between the map and the real world. Both qualitatively, in terms of overall structure, and quantitatively, in terms of geometric precision, a highly abstract map does not compare favorably to a high-fidelity map.

Despite this disadvantage, decomposition and abstraction may be useful if the abstraction can be planned carefully so as to capture the relevant, *useful* features of the world while discarding all other features. The advantage of this approach is that the map representation can potentially be minimized. Furthermore, if the decomposition is hierarchical, such as in a pyramid of recursive abstraction, then reasoning and planning with respect to the map representation may be computationally far superior to planning in a fully detailed world model.

A standard, lossless form of *opportunistic decomposition* is termed *exact cell decomposition.* This method, introduced by [5], achieves decomposition by selecting boundaries between discrete cells based on geometric criticality.

Figure 5.14 depicts an exact decomposition of a planar workspace populated by polygonal obstacles. The map representation tessellates the space into areas of free space. The representation can be extremely compact because each such area is actually stored as a single node, shown in the graph at the bottom of Figure 5.14.

The underlying assumption behind this decomposition is that the particular position of a ro-

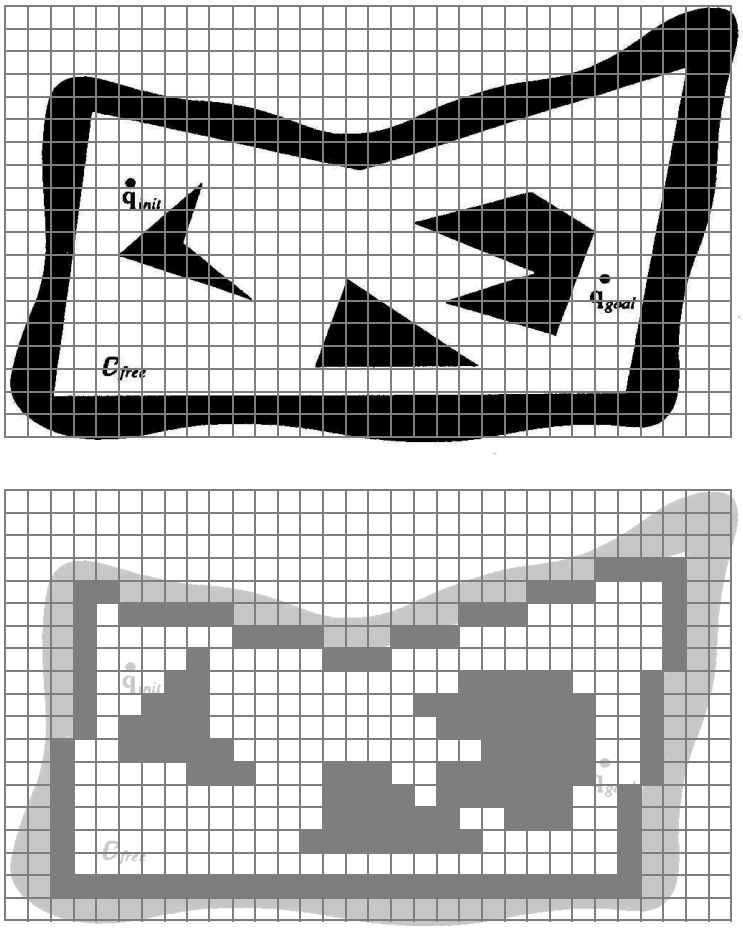


Fig 5.15 Fixed decomposition of the same space. (narrow passage disappears)

bot within each area of free space does not matter. What matters is the robot's ability to traverse from each area of free space to the adjacent areas. Therefore, as with other representations we will see, the resulting graph captures the adjacency of map locales. If indeed the assumptions are valid and the robot does not care about its precise position within a single area, then this can be an effective representation that nonetheless captures the connectivity of the environment.

Such an exact decomposition is not always appropriate. Exact decomposition is a function of the particular environment obstacles and free space. If this information is expensive to collect or even unknown, then such an approach is not feasible.

An alternative is *fixed decomposition*, in which the world is tessellated, transforming the continuos real environment into a discrete approximation for the map. Such a transformation is demonstrated in Figure 5.15, which depicts what happens to obstacle-filled and free areas during this transformation. The key disadvantage of this approach stems from its *inexact* nature. It is possible for narrow passageways to be lost during such a transformation, as shown in Figure 5.15. Formally this means that fixed decomposition is sound but not complete. Yet another approach is adaptive cell decomposition as presented in Figure 5.16.

The concept of fixed decomposition is extremely popular in mobile robotics; it is perhaps the single most common map representation technique currently utilized. One very popular

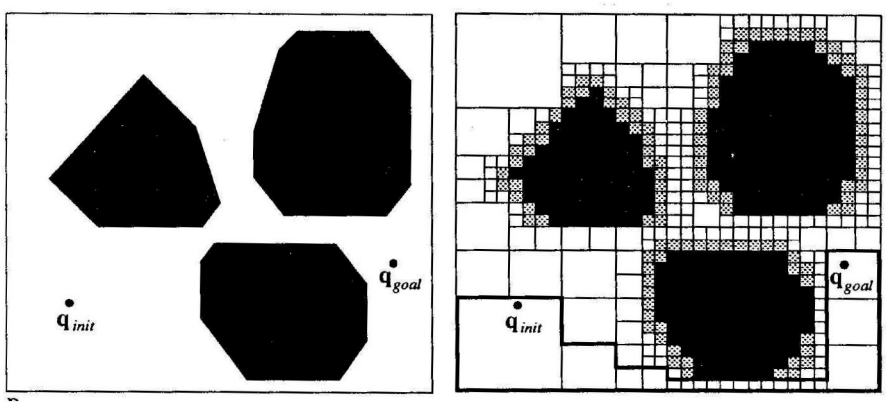


Fig 5.16 *Example of adaptive decomposition of an environment.*

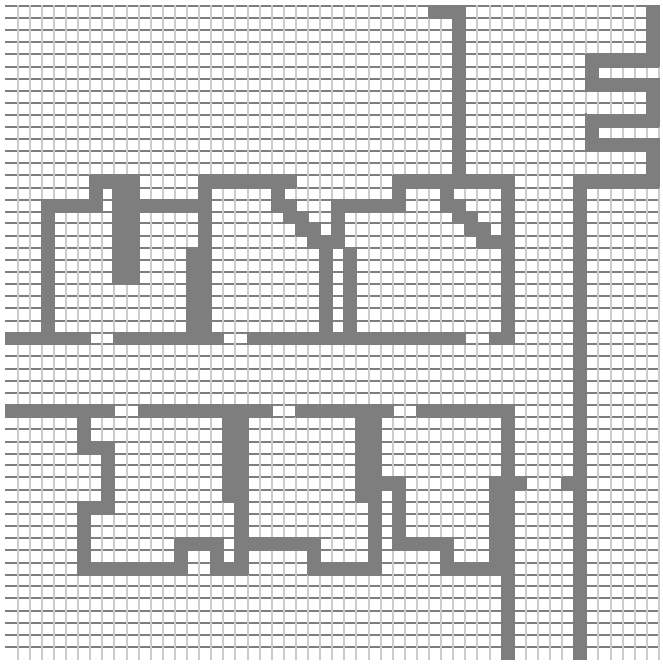


Fig 5.17 *Example of an occupancy grid map representation.*

version of fixed decomposition is known as the *occupancy grid* representation [91]. In an occupancy grid, the environment is represented by a discrete grid, where each cell is either filled (part of an obstacle) or empty (part of free space). This method is of particular value when a robot is equipped with range-based sensors because the range values of each sensor, combined with the absolute position of the robot, can be used directly to update the filled/ empty value of each cell.

In the occupancy grid, each cell may have a counter, whereby the value 0 indicates that the cell has not been "hit" by any ranging measurements and, therefore, it is likely free space. As the number of ranging strikes increases, the cell's value is incremented and, above a certain threshold, the cell is deemed to be an obstacle. By discounting the values of cells over time, both hysteresis and the possibility of transient obstacles can be represented using this occupancy grid approach. Figure 5.17 depicts an occupancy grid representation in which the darkness of each cell is proportional to the value of its counter. One commercial robot that uses a standard occupancy grid for mapping and navigation is the Cye robot [112].

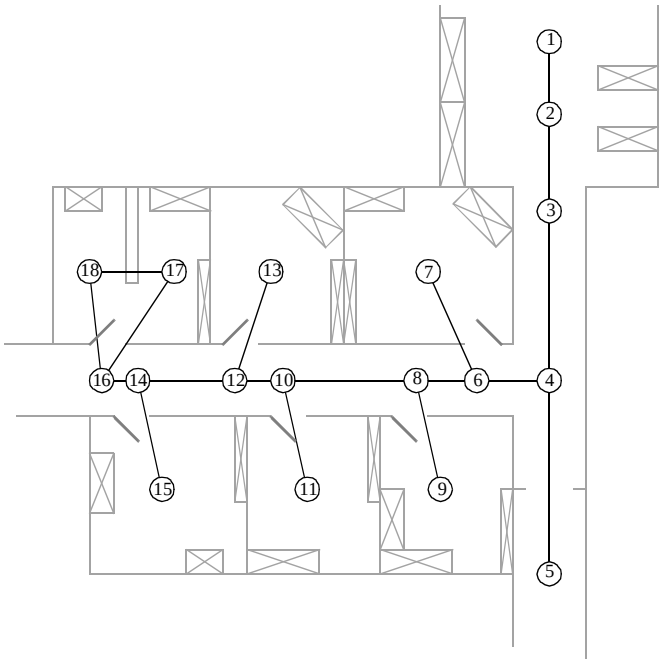


Fig 5.18 *A topological representation of an indoor office area.*

There remain two main disadvantages of the occupancy grid approach. First, the size of the map in robot memory grows with the size of the environment and, if a small cell size is used, this size can quickly become untenable. This occupancy grid approach is not compatible with the *closed world assumption*, which enabled continuous representations to have potentially very small memory requirements in large, sparse environments. In contrast, the occupancy grid must have memory set aside for every cell in the matrix. Furthermore, any fixed decomposition method such as this imposes a geometric grid on the world *a priori*, regardless of the environmental details. This can be inappropriate in cases where geometry is not the most salient feature of the environment.

For these reasons, an alternative, called *topological* decomposition, has been the subject of some exploration in mobile robotics. Topological approaches avoid direct measurement of geometric environmental qualities, instead concentrating on characteristics of the environment that are most relevant to the robot for localization.

Formally, a topological representation is a graph that specifies two things: *nodes* and the *connectivity* between those nodes. Insofar as a topological representation is intended for the use of a mobile robot, nodes are used to denote areas in the world and arcs are used to denote adjacency of pairs of nodes. When an arc connects two nodes, then the robot can traverse from one node to the other without requiring traversal of any other intermediary node.

Adjacency is clearly at the heart of the topological approach, just as adjacency in a cell decomposition representation maps to geometric adjacency in the real world. However, the topological approach diverges in that the nodes are not of fixed size nor even specifications of free space. Instead, nodes document an area based on any sensor discriminant such that the robot can recognize entry and exit of the node.

Figure 5.18 depicts a topological representation of a set of hallways and offices in an indoor

environment. In this case, the robot is assumed to have an intersection detector, perhaps using sonar and vision to find intersections between halls and between halls and rooms. Note that nodes capture geometric space and arcs in this representation simply represent connectivity.

Another example of topological representation is the work of Dudek [49], 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 Dudek's 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 the potential to be both compact like a topological representation and precise as with all direct geometric representations.

Yet, a chief motivation of the topological approach is that the environment may contain important non-geometric features - features that have no ranging relevance but are useful for localization. In Chapter 4 we described such whole-image vision-based features.

In contrast to these whole-image feature extractors, often spatially localized landmarks are artificially placed in an environment to impose a particular visual-topological connectivity upon the environment. In effect, the artificial landmark can impose artificial structure. Examples of working systems operating with this landmark-based strategy have also demonstrated success. Latombe's landmark-based navigation research [89] has been implemented on real-world indoor mobile robots that employ paper landmarks attached to the ceiling as the locally observable features. Chips the museum robot is another robot that uses man-made landmarks to obviate the localization problem. In this case, a bright pink square serves as a landmark with dimensions and color signature that would be hard to accidentally reproduce in a museum environment [88]. One such museum landmark is shown in Figure (5.19).

In summary, range is clearly not the only measurable and useful environmental value for a mobile robot. This is particularly true due to the advent of color vision as well as laser rangefinding, which provides reflectance information in addition to range information. Choosing a map representation for a particular mobile robot requires first understanding the sensors available on the mobile robot and second understanding the mobile robot's functional requirements (*e.g.* required goal precision and accuracy).

5.5.3 State of the Art: Current Challenges in Map Representation

The sections above describe major design decisions in regards to map representation choic-

Fig 5.19 An artificial landmark used by Chips during autonomous docking.

es. There are, however, fundamental real-world features that mobile robot map representations do not yet represent well. These continue to be the subject of open research, and several such challenges are described below.

The real world is dynamic. As mobile robots come to inhabit the same spaces as humans, they will encounter moving people, cars, strollers and the transient obstacles placed and moved by humans as they go about their activities. This is particularly true when one considers the home environment with which domestic robots will someday need to contend.

The map representations described above do not, in general, have explicit facilities for identifying and distinguishing between permanent obstacles (e.g. walls, doorways, etc.) and transient obstacles (e.g. humans, shipping packages, etc.). The current state of the art in terms of mobile robot sensors is partly to blame for this shortcoming. Although vision research is rapidly advancing, robust sensors that discriminate between moving animals and static structures *from a moving reference frame* are not yet available. Furthermore, estimating the motion vector of transient objects remains a research problem.

Usually, the assumption behind the above map representations is that all objects on the map are effectively static. Partial success can be achieved by discounting mapped objects over time. For example, occupancy grid techniques can be more robust to dynamic settings by introducing temporal discounting, effectively treating transient obstacles as noise. The more challenging process of map creation is particularly fragile to environment dynamics; most mapping techniques generally require that the environment be free of moving objects during the mapping process. One exception to this limitation involves topological representations. Because precise geometry is not important, transient objects have little effect on the mapping or localization process, subject to the critical constraint that the transient objects must not change the topological connectivity of the environment. Still, neither the occupancy grid representation nor a topological approach is actively recognizing and representing transient

objects as distinct from both sensor error and permanent map features.

As vision sensing provides more robust and more informative content regarding the transience and motion details of objects in the world, mobile roboticists will in time propose representations that make use of that information. A classic example involves occlusion by human crowds. Museum tour guide robots generally suffer from an extreme amount of occlusion. If the robot's sensing suite is located along the robot's body, then the robot is effectively blind when a group of human visitors completely surrounds the robot. This is because its map contains only environment features that are, at that point, fully hidden from the robot's sensors by the wall of people. In the best case, the robot should recognize its occlusion and make no effort to localize using these invalid sensor readings. In the worst case, the robot will localize with the fully occluded data, and will update its location incorrectly. A vision sensor that can discriminate the local conditions of the robot (*e.g. we are surrounded by people*) can help eliminate this error mode.

A second open challenge in mobile robot localization involves the traversal of open spaces. Existing localization techniques generally depend on local measures such as range, thereby demanding environments that are somewhat densely filled with objects that the sensors can detect and measure. Wide open spaces such as parking lots, fields of grass and indoor atriums such as those found in convention centers pose a difficulty for such systems due to their relative sparseness. Indeed, when populated with humans, the challenge is exacerbated because any mapped objects are almost certain to be occluded from view by the people.

Once again, more recent technologies provide some hope for overcoming these limitations. Both vision and state-of-the-art laser rangefinding devices offer outdoor performance with ranges of up to a hundred meters and more. Of course, GPS performs even better. Such long-range sensing may be required for robots to localize using distant features.

This trend teases out a hidden assumption underlying most topological map representations. Usually, topological representations make assumptions regarding spatial locality: a node contains objects and features that are themselves within that node. The process of map creation thus involves making nodes that are, in their own self-contained way, recognizable by virtue of the objects contained within the node. Therefore, in an indoor environment, each room can be a separate node, and this is reasonable because each room will have a layout and a set of belongings that are unique to that room.

However, consider the outdoor world of a wide-open park. Where should a single node end and the next node begin? The answer is unclear because objects that are far away from the current node, or position, can yield information for the localization process. For example, the hump of a hill at the horizon, the position of a river in the valley and the trajectory of the sun all are non-local features that have great bearing on one's ability to infer current position. The spatial locality assumption is violated and, instead, replaced by a visibility criterion: the node or cell may need a mechanism for representing objects that are measurable and visible from that cell. Once again, as sensors improve and, in this case, as outdoor locomotion mechanisms improve, there will be greater urgency to solve problems associated with localization in wide-open settings, with and without GPS-type global localization sensors.

We end this section with one final open challenge that represents one of the fundamental academic research questions of robotics: sensor fusion. A variety of measurement types are possible using off-the-shelf robot sensors, including heat, range, acoustic and light-based reflectivity, color, texture, friction, etc. Sensor fusion is a research topic closely related to map representation. Just as a map must embody an environment in sufficient detail for a robot to perform localization and reasoning, sensor fusion demands a representation of the world that is sufficiently general and expressive that a variety of sensor types can have their data correlated appropriately, strengthening the resulting percepts well beyond that of any individual sensor's readings.

Perhaps the only general implementation of sensor fusion to date is that of neural network classifier. Using this technique, any number and any type of sensor values may be jointly combined in a network that will use whatever means necessary to optimize its classification accuracy. For the mobile robot that must use a human-readable internal map representation, no equally general sensor fusion scheme has yet been born. It is reasonable to expect that, when the sensor fusion problem is solved, integration of a large number of disparate sensor types may easily result in sufficient discriminatory power for robots to achieve real-world navigation, even in wide-open and dynamic circumstances such as a public square filled with people.

5.6 Probabilistic Map-Based Localization

5.6.1 Introduction

As stated earlier, multiple hypothesis position representation is advantageous because the robot can explicitly track its own beliefs regarding its possible positions in the environment. Ideally, the robot's *belief state* will change, over time, as is consistent with its motor outputs and perceptual inputs. One geometric approach to multiple hypothesis representation, mentioned earlier, involves identifying the possible positions of the robot by specifying a polygon in the environmental representation [113]. This method does not provide any indication of the relative chances between various possible robot positions.

Probabilistic techniques differ from this because they explicitly identify probabilities with the possible robot positions, and for this reason these methods have been the focus of recent research. In the following sections we present two classes of probabilistic localization. The first class, *Markov localization*, uses an explicitly specified probability distribution across all possible robots positions. The second method, *Kalman filter localization*, uses a Gaussian probability density representation of robot position and scan matching for localization. Unlike Markov localization, Kalman filter localization does not independently consider each possible pose in the robot's configuration space. Interestingly, the Kalman filter localization process results from the Markov localization axioms if the robot's position uncertainty is assumed to have a Gaussian form [28 page 43-44].

Before discussing each method in detail, we present the general robot localization problem and solution strategy. Consider a mobile robot moving in a known environment. As it starts to move, say from a precisely known location, it can keep track of its motion using odometry. Due to odometry uncertainty, after some movement the robot will become very uncertain about its position (see section 5.2.4). To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot might use its on-board sensors (ultrasonic, range sensor, vision) to make observations of its environment. The information provided by the robot's odometry, plus the information provided by such exteroceptive observations can be combined to enable the robot to localize as well as possible with respect to its map. The processes of updating based on proprioceptive sensor values and exteroceptive sensor values are often separated logically, leading to a general two-step process for robot position update.

Action update represents the application of some action model *Act* to the mobile robot's proprioceptive encoder measurements o_t and prior belief state s_{t-1} to yield a new belief state representing the robot's belief about its current position. Note that throughout this chapter we will assume that the robot's proprioceptive encoder measurements are used as the best possible measure of its actions over time. If, for instance, a differential drive robot had motors without encoders connected to its wheels and employed open-loop control, then instead of encoder measurements the robot's highly uncertain estimates of wheel spin would need to be incorporated. We ignore such cases and therefore have a simple formula:

$$s'_t = Act(o_t, s_{t-1}). \tag{5.16}$$

Perception update represents the application of some perception model *See* to the mobile robot's exteroceptive sensor inputs i_t and updated belief state s'_t to yield a refined belief state representing the robot's current position:

$$s_t = See(i_t, s'_t) \tag{5.17}$$

The perception model *See* and sometimes the action model *Act* are abstract functions of both the map and the robot's physical configuration (e.g. sensors and their positions, kinematics, etc.).

In general, the action update process contributes uncertainty to the robot's belief about position: encoders have error and therefore motion is somewhat nondeterministic. By contrast, perception update generally refines the belief state. Sensor measurements, when compared to the robot's environmental model, tend to provide clues regarding the robot's possible position.

In the case of Markov localization, the robot's belief state is usually represented as separate probability assignments for every possible robot pose in its map. The action update and perception update processes must update the probability of every cell in this case. Kalman filter localization represents the robot's belief state using a singe, well-defined Gaussian probability density function, and thus retains just a μ and σ parameterization of the robot's belief about position with respect to the map. Updating the parameters of the Gaussian distribution is all that is required. This fundamental difference in the representation of belief state leads to the following advantages and disadvantages of the two methods, as presented in [44]:

- Markov localization allows for localization starting from any unknown position and can thus recover from ambiguous situations because the robot can track multiple, completely disparate possible positions. However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and computational power can thus limit precision and map size.
- Kalman filter localization tracks the robot from an initially known position and is inherently both precise and efficient. In particular, Kalman filter localization can be used in continuous world representations. However, if the uncertainty of the robot becomes too large (e.g. due to a robot collision with an object) and thus not truly unimodal, the Kalman filter can fail to capture the multitude of possible robot positions and can become irrevocably lost.

In recent research projects improvements are achieved or proposed by either only updating the state space of interest within the Markov approach [43] or by combining both methods to create a hybrid localization system [44]. In the next two subsections we will present each approach in detail.

5.6.2 Markov Localization (see also [42, 45, 71, 72])

Markov localization tracks the robot's belief state using an arbitrary probability density function to represent the robot's position. In practice, all known Markov localization sys-

tems implement this generic belief representation by first tessellating the robot configuration space into a finite, discrete number of possible robot poses in the map. In actual applications, the number of possible poses can range from several hundred positions to millions of positions.

Given such a generic conception of robot position, a powerful update mechanism is required that can compute the belief state that results when new information (e.g. encoder values and sensor values) is incorporated into a prior belief state with arbitrary probability density. The solution is born out of probability theory, and so the next section describes the foundations of probability theory that apply to this problem, notably Bayes formula. Then, two subsequent subsections provide case studies, one robot implementing a simple feature-driven topological representation of the environment [45, 71, 72] and the other using a geometric grid-based map[42, 43].

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 B 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 Bayes formula for computing $p(A|B)$:

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

We use 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 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 $p(r=l)$:

$$p(l|i) = \frac{p(i|l)p(l)}{p(i)} \tag{5.21}$$

The value of $p(i|l)$ is key to Equation 5.21, and this probability of a sensor input at each robot position must be computed using some model. An obvious strategy would be to consult the robot's map, identifying the probability of particular sensor readings with each possible map position, given knowledge about the robot's sensor geometry and the mapped environment. The value of $p(l)$ is easy to recover in this case. It is simply the probability $p(r=l)$ associated with the belief state before the perceptual update process. Finally, note that the denominator $p(i)$ does not depend upon l; that is, as we apply Equation 5.21 to all positions l in L, the denominator never varies. Because it is effectively constant, in practice this denominator is usually dropped and, at the end of the perception update step, all probabilities in the belief state are re-normalized to sum at 1.0.

Now consider the *Act* function of Equation 5.16. *Act* maps a former belief state and encoder measurement (i.e. robot action) to a new belief state. In order to compute the probability of position l in the new belief state, one must integrate over all the possible ways in which the robot may have reached l according to the potential positions expressed in the former belief state. This is subtle but fundamentally important. The same location l can be reached from multiple source locations with the same encoder measurement o because the encoder measurement is uncertain. Temporal indices are required in this update equation:

$$p(l_t|o_t) = \int p(l_t|l'_{t-1}, o_t)p(l'_{t-1})dl'_{t-1} \tag{5.22}$$

Thus, the total probability for a specific position l is built up from the individual contributions from every location l' in the former belief state given encoder measurement o.

Equations 5.21 and 5.22 form the basis of Markov localization, and they incorporate the *Markov assumption.* Formally, this means that their output is a function only of the robot's previous state and its most recent actions (odometry) and perception. In a general, non-Markovian situation, the state of a system depends upon all of its history. After all, the value of a robot's sensors at time t do not really depend only on its position at time t. They depend to some degree on the trajectory of the robot over time; indeed on the entire history of the robot. For example, the robot could have experienced a serious collision recently that has biased the sensor's behavior. By the same token, the position of the robot at time t does not

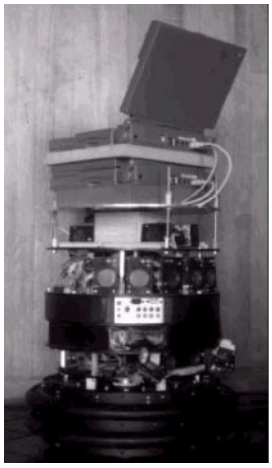


Fig 5.20 Dervish exploring its environment

really depend only on its position at time *t-1* and its odometric measurements. Due to its history of motion, one wheel may have worn more than the other, causing a left-turning bias over time that affects its current position.

So the Markov assumption is, of course, not a valid assumption. However the Markov assumption greatly simplifies tracking, reasoning and planning and so it is an approximation that continues to be extremely popular in mobile robotics.

5.6.2.2 Case Study I: Markov Localization using a Topological Map

A straightforward application of Markov localization is possible when the robot's environment representation already provides an appropriate decomposition. This is the case when the environment representation is purely topological.

Consider a contest in which each robot is to receive a topological description of the environment. The description would describe only the connectivity of hallways and rooms, with no mention of geometric distance. In addition, this supplied *map* would be imperfect, containing several false arcs (e.g. a closed door). Such was the case for the 1994 AAAI National Robot Contest, at which each robot's mission was to use the supplied map and its own sensors to navigate from a chosen starting position to a target room.

Dervish, the winner of this contest, employed probabilistic Markov localization and used just this multiple hypothesis belief state over a topological environmental representation. We now describe Dervish as an example of a robot with a topological representation and a probabilistic localization algorithm.

Dervish, shown in Figure 5.20, includes a sonar arrangement custom-designed for the 1994 AAAI National Robot Contest. The environment in this contest consisted of a rectilinear indoor office space filled with real office furniture as obstacles. Traditional sonars are arranged radially around the robot in a ring. Robots with such sensor configurations are subject to both tripping over short objects below the ring and to decapitation by tall objects (such as ledges, shelves and tables) that are above the ring.

Dervish's answer to this challenge was to arrange one pair of sonars diagonally upward to

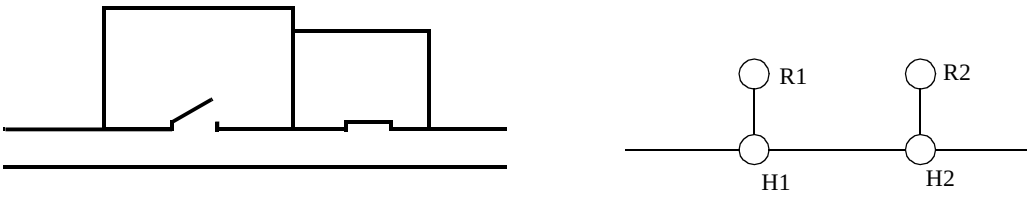


Fig 5.21 A geometric office environment (left) and its topological analogue (right)

detect ledges and other overhangs. In addition, the diagonal sonar pair also proved to ably detect tables, enabling the robot to avoid wandering underneath tall tables. The remaining sonars were clustered in sets of sonars, such that each individual transducer in the set would be at a slightly varied angle to minimize specularity. Finally, two sonars near the robot's base were able to detect low obstacles such as paper cups on the floor.

We have already noted that the representation provided by the contest organizers was purely topological, noting the connectivity of hallways and rooms in the office environment. Thus, it would be appropriate to design Dervish's perceptual system to detect matching perceptual events: the detection and passage of connections between hallways and offices.

This *abstract* perceptual system was implemented by viewing the trajectory of sonar strikes to the left and right sides of Dervish over time. Interestingly, this perceptual system would use time alone and no concept of encoder value in order to trigger perceptual events. Thus, for instance, when the robot detects a 7 to 17 cm indentation in the width of the hallway for more than one second continuously, a *closed door* sensory event is triggered. If the sonar strikes jump well beyond 17 cm for more than one second, an *open door* sensory event triggers.

Sonars have a notoriously problematic error mode known as specular reflection: when the sonar unit strikes a flat surface at a shallow angle, the sound may reflect coherently away from the transducer, resulting in a large overestimate of range. Dervish was able to filter such potential noise by tracking its approximate angle in the hallway and completely suppressing sensor events when its angle to the hallway parallel exceeded 9 degrees. Interestingly, this would result in a conservative perceptual system that would easily miss features because of this suppression mechanism, particularly when the hallway is crowded with obstacles that Dervish must negotiate. Once again, the conservative nature of the perceptual system, and in particular its tendency to issue false negatives, would point to a probabilistic solution to the localization problem so that a complete trajectory of perceptual inputs could be considered.

Dervish's environment representation was a classical topological map, identical in abstraction and information to the map provided by the contest organizers. Figure 5.21 depicts a geometric representation of a typical office environment and the topological map for the same office environment. One can place nodes at each intersection and in each room, resulting in the case of figure 5.21 with four nodes total.

Once again, though, it is crucial that one maximize the information content of the representation based on the available percepts. This means reformulating the standard topological graph shown in Figure 5.21 so that transitions *into* and *out of* intersections may both be used

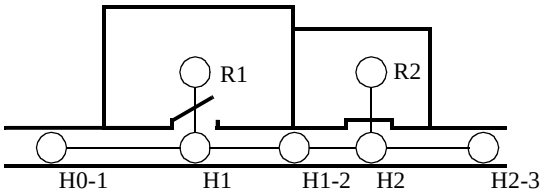


Fig 5.22 *A modification of the topological map to maximize information.*

for position updates. Figure 5.22 shows a modification of the topological map in which just this step has been taken. In this case, note that there are 7 nodes in contrast to 4.

In order to represent a specific belief state, Dervish associated with each topological node *n* a probability that the robot is at a physical position within the boundaries of *n*: $p(r_t = n)$.

As will become clear below, the probabilistic update used by Dervish was approximate, therefore technically one should refer to the resulting values as *likelihoods* rather than probabilities.

The perception update process for Dervish functions precisely as in Equation (5.21). Perceptual events are generated asynchronously, each time the feature extractor is able to recognize a large-scale feature (e.g. doorway, intersection) based on recent ultrasonic values. Each perceptual event consists of a percept-pair (a feature on one side of the robot or two features on both sides).

Table 5.1: Dervish's certainty matrix.

	Wall	*Closed Door*	*Open Door*	*Open Hallway*	*Foyer*
Nothing detected	*0.70*	*0.40*	*0.05*	*0.001*	*0.30*
Closed door detected	*0.30*	*0.60*	*0*	*0*	*0.05*
Open door detected	*0*	*0*	*0.90*	*0.10*	*0.15*
Open hallway detected	*0*	*0*	*0.001*	*0.90*	*0.50*

Given a specific percept pair *i*, Equation (5.21) enables the likelihood of each possible position *n* to be updated using the formula:

$$p(n|i) = p(i|n)p(n) \tag{5.23}$$

The value of $p(n)$ is already available from the current belief state of Dervish, and so the challenge lies in computing $p(i|n)$. The key simplification for Dervish is based upon the realization that, because the feature extraction system only extracts 4 total features and because a node contains (on a single side) one of 5 total features, every possible combination of node type and extracted feature can be represented in a 4 x 5 table.

Dervish's *certainty matrix* (show in Table 5.1) is just this lookup table. Dervish makes the simplifying assumption that the performance of the feature detector (i.e. the probability that it is correct) is only a function of the feature extracted and the actual feature in the node. With this assumption in hand, we can populate the *certainty matrix* with confidence esti-

mates for each possible pairing of perception and node type. For each of the five world features that the robot can encounter (wall, closed door, open door, open hallway and foyer) this matrix assigns a likelihood for each of the three one-sided percepts that the sensory system can issue. In addition, this matrix assigns a likelihood that the sensory system will fail to issue a perceptual event altogether (*nothing detected*).

For example, using the specific values in Table 5.1, if Dervish is next to an open hallway, the likelihood of mistakenly recognizing it as an open door is 0.10. This means that for any node n that is of type *Open Hallway* and for the sensor value i=*Open door*, $p(i|n) = 0.10$. Together with a specific topological map, the certainty matrix enables straightforward computation of $p(i|n)$ during the perception update process.

For Dervish's particular sensory suite and for any specific environment it intends to navigate, humans generate a specific certainty matrix that loosely represents its perceptual confidence, along with a global measure for the probability that any given door will be closed versus opened in the real world.

Recall that Dervish has no encoders and that perceptual events are triggered asynchronously by the feature extraction processes. Therefore, Dervish has no action update step as depicted by Equation (5.22). When the robot does detect a perceptual event, multiple perception update steps will need to be performed in order to update the likelihood of every possible robot position given Dervish's former belief state. This is because there is often a chance that the robot has traveled *multiple* topological nodes since its previous perceptual event (i.e. false negative errors). Formally, the perception update formula for Dervish is in reality a combination of the general form of action update and perception update. The likelihood of position n given perceptual event i is calculated as in Equation (5.22):

$$p(n_t|i_t) = \int p(n_t|n'_{t-i}, i_t)p(n'_{t-i})dn'_{t-i} \tag{5.24}$$

The value of $p(n'_{t-i})$ denotes the likelihood of Dervish being at position n' as represented by Dervish's former belief state. The temporal subscript *t-i* is used in lieu of *t-1* because for each possible position n' the discrete topological distance from n' to n can vary depending on the specific topological map. The calculation of $p(n_t|n'_{t-i}, i_t)$ is performed by multiplying the probability of generating perceptual event i at position n by the probability of having failed to generate perceptual events at all nodes between n' and n:

$$p(n_t|n'_{t-i}, i_t) = p(i_t, n_t) \cdot p(\varnothing, n_{t-1}) \cdot p(\varnothing, n_{t-2}) \cdot \ldots \cdot p(\varnothing, n_{t-i+1}) \tag{5.25}$$

For example (figure 5.23), suppose that the robot has only two nonzero nodes in its belief state, {1-2, 2-3}, with likelihoods associated with each possible position: p(1-2) = 1.0 and p(2-3) = 0.2. For simplicity assume the robot is facing East with certainty. Note that the likelihoods for nodes 1-2 and 2-3 do not sum to 1.0. These values are not formal probabilities, and so computational effort is minimized in Dervish by avoiding normalization altogether. Now suppose that a perceptual event is generated: the robot detects an open hallway

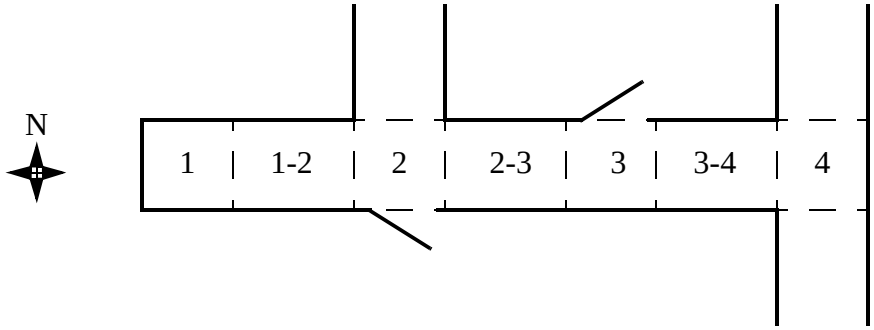


Fig 5.23 *A realistic indoor topological environment.*

on its left and an open door on its right simultaneously.

State 2-3 will progress potentially to states 3, 3-4 and 4. But states 3 and 3-4 can be eliminated because the likelihood of detecting an open door when there is only wall is zero. The likelihood of reaching state 4 is the product of the initial likelihood for state 2-3, 0.2, the likelihood of not detecting anything at node 3, (a), and the likelihood of detecting a hallway on the left and a door on the right at node 4, (b). Note that we assume the likelihood of detecting nothing at node 3-4 is 1.0 (a simplifying approximation).

(a) occurs only if Dervish fails to detect the door on its left at node 3 (either closed or open), [(0.6)(0.4) + (1-0.6)(0.05)], and correctly detects nothing on its right, 0.7.

(b) occurs if Dervish correctly identifies the open hallway on its left at node 4, 0.90, and mistakes the right hallway for an open door, 0.10.

The final formula, (0.2)[(0.6)(0.4)+(0.4)(0.05)](0.7)[(0.9)(0.1)], yields a likelihood of 0.003 for state 4. This is a partial result for $p(4)$ following from the prior belief state node 2-3.

Turning to the other node in Dervish's prior belief state, 1-2 will potentially progress to states 2, 2-3, 3, 3-4 and 4. Again, states 2-3, 3 and 3-4 can all be eliminated since the likelihood of detecting an open door when a wall is present is zero. The likelihood of state 2 is the product of the prior likelihood for state 1-2, (1.0), the likelihood of detecting the door on the right as an open door, [(0.6)(0) + (0.4)(0.9)], and the likelihood of correctly detecting an open hallway to the left, 0.9. The likelihood for being at state 2 is then (1.0)(0.4)(0.9)(0.9) = 0.3. In addition, 1-2 progresses to state 4 with a certainty factor of $4.3 \cdot 10^{-6}$, which is added to the certainty factor above to bring the total for state 4 to 0.00328. Dervish would therefore track the new belief state to be {2, 4}, assigning a very high likelihood to position 2 and a low likelihood to position 4.

Empirically, Dervish's map representation and localization system have proven to be sufficient for navigation of four indoor office environments: the artificial office environment created explicitly for the 1994 National Conference on Artificial Intelligence; the psychology department, the history department and the computer science department at Stanford University. All of these experiments were run while providing Dervish with no notion of the distance between adjacent nodes in its topological map. It is a demonstration of the power of probabilistic localization that, in spite of the tremendous lack of action and encoder information, the robot is able to navigate several real-world office buildings successfully.

One open question remains with respect to Dervish's localization system. Dervish was not just a localizer but also a navigator. As with all multiple hypothesis systems, one must ask the question, how does the robot decide how to move, given that it has multiple possible robot positions in its representation? The technique employed by Dervish is a most common technique in the mobile robotics field: plan the robot's actions by assuming that the robot's actual position is its most likely node in the belief state. Generally, the most likely position is a good measure of the robot's actual world position. However, this technique has shortcomings when the highest and second highest most likely positions have similar values. In the case of Dervish, it nonetheless goes with the highest likelihood position at all times, save at one critical juncture. The robot's goal is to enter a target room and remain there. Therefore, from the point of view of its goal, it is critical that it finish navigating only when the robot has strong confidence in being at the correct final location. In this particular case, Dervish's execution module refuses to enter a room if the gap between the most likely position and the second likeliest position is below a preset threshold. In such a case, Dervish will actively plan a path that causes it to move further down the hallway in an attempt to collect more sensor data and thereby increase the relative likelihood of one position in the belief state.

Although computationally unattractive, one can go further, imagining a planning system for robots such as Dervish for which one specifies a *goal belief state* rather than a goal position. The robot can then reason and plan in order to achieve a goal confidence level, thus explicitly taking into account not only robot position but also the measured likelihood of each position. An example of just such a procedure is the Sensory Uncertainty Field of Latombe [90], in which the robot must find a trajectory that reaches its goal while maximizing its localization confidence enroute.

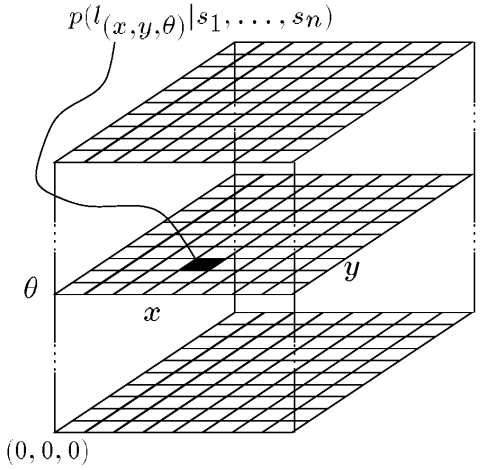


Fig 5.24 The belief state representation 3D array used by Rhino

5.6.2.3 Case Study II: 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 et al.* [42, 43] 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 24 sonars and 2 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 this 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 - 64cm in various instantiations.

Like Dervish, Rhino uses multiple hypothesis belief representation. In line with the far improved resolution of the environment representation, the belief state representation of Rhino consists of a 15 x 15 x 15 3D array representing the probability of 15^3 possible robot positions (see Figure 5.24). The resolution of the array is 15cm x 15cm x 1°. 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 environment representation in order for the overall system to function well.

Whereas Dervish made use 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:

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

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 3-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 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 $p(i|l)$ directly using a model of the robot's sensor behavior, its position l and the local environmental metric map around l.

The sensor model must calculate the probability of a specific perceptual measurement given that its likelihood is justified by known errors of the sonar or laser rangefinder sensors. Three key assumptions are used to construct this sensor model:

1 *If an object in the metric map is detected by a range sensor, the measurement error can be described with a distribution that has a mean at the correct reading.*

2 *There should always be a nonzero chance that a range sensor will read any measurement value, even if this measurement disagrees sharply with the environmental geometry.*

3 *In contrast to the generic error described in #2, there is a specific failure mode in ranging sensors whereby the signal is absorbed or coherently reflected, causing the sensor's range measurement to be maximal. Therefore, there is a local peak in the probability density distribution at the maximal reading of a range sensor.*

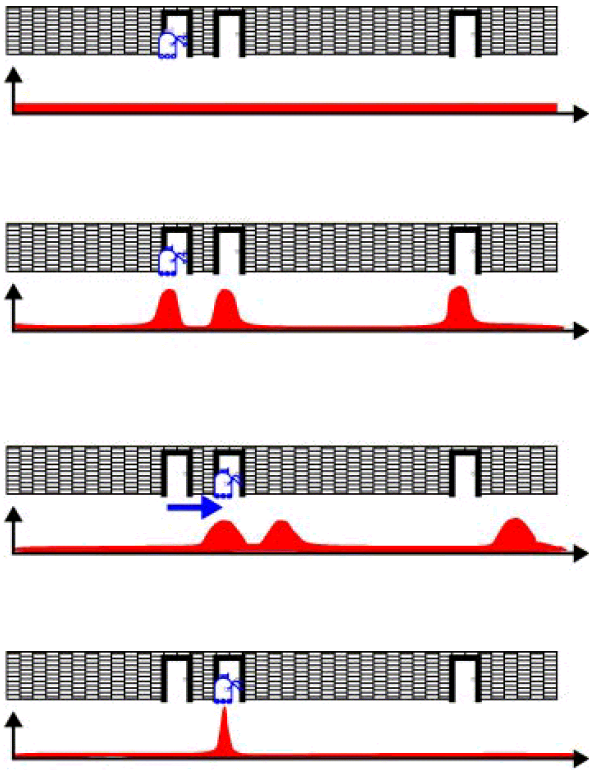


Fig 5.25 *Improving belief state by moving. Roland, we need to re-make this picture for copyright reasons probably.*

By validating these assumptions using empirical sonar trials in multiple environments, the research group has delivered to Rhino a conservative and powerful sensor model for its particular sensors.

Figure 5.25 provides a simple 1D example of the grid-based Markov localization algorithm. The robot begins with a flat probability density function for its possible location. In other words, it initially has no bias regarding position. As the robot encounters first one door and then a second door, the probability density function over possible positions becomes first multimodal and finally unimodal and sharply defined. The ability of a Markov localization system to *localize* the robot from an initially lost belief state is its key distinguishing feature.

The resulting robot localization system has been part of a navigation system that has demonstrated great success both at the University of Bonn and at a public museum in Bonn. This is a challenging application because of the dynamic nature of the environment, as the robot's sensors are frequently subject to occlusion due to humans gathering around the robot. Rhino's ability to function well in this setting is a demonstration of the power of the Markov localization approach

Reducing computational complexity: Randomized Sampling

A great many steps are taken in real-world implementations such as Rhino in order to effect computational gains. These are valuable because, with an exact cell decomposition representation and use of raw sensor values rather than abstraction to features, such a robot has a

massive computational effort associated with each perceptual update.

One class of techniques deserves mention because it can significantly reduce the computational overhead of techniques that employ fixed-cell decomposition representations. The basic idea, which we call *randomized sampling* is known alternatively as Particle filter algorithms, Condensation algorithms and Monte Carlo algorithms [ROLAND, reference the Thrun et al. paper on "Robust Monte Carlo" in my text file].

Irrespective of the specific technique, the basic algorithm is the same in all these cases. Instead of representing *every* possibe robot position by representing the complete and correct belief state, an approximate belief state is constructed by representing only a *subset* of the complete set of possible locations that should be considered.

For example, consider a robot with a complete belief state of 10,000 possible locations at time *t.* Instead of tracking and updating all 10,000 possible locations based on a new sensor measurement, the robot can select only 10% of the stored locations and update only those locations. By weighting this sampling process with the probability values of the locations, one can bias the system to generate more samples at local peaks in the probability density function. So, the resulting 1,000 locations will be concentrated primarily at the highest probability locations. This biasing is desirable, but only to a point.

We also wish to ensure that *some* less likely locations are tracked, as otherwise, if the robot does indeed receive unlikely sensor measurements, it will fail to localize. This *randomization* of the sampling process can be performed by adding additional samples from a flat distribution for example. Further enhancements of these randomized methods enable the number of statistical samples to be varied on-the-fly, based for instance on the ongoing localization confidence of the system. This further reduces the number of samples required on average while guaranteeing that a large number of samples will be used when necessary [ROLAND, reference the Fox NIPS article on KLD-Sampling Adaptive Particle Filters]

These sampling techniques have resulted in robots that function indistinguishably as compared to their full belief state set ancestors, yet use computationally a fraction of the resources. Of course, such sampling has a penalty: completeness. The probabilistically complete nature of Markov localization is violated by these sampling approaches because the robot is failing to update *all* the nonzero probability locations, and thus there is a danger that the robot, due to an unlikely but correct sensor reading, could become truly lost. Of course, recovery from a lost state is feasible just as with all Markov localization techniques.

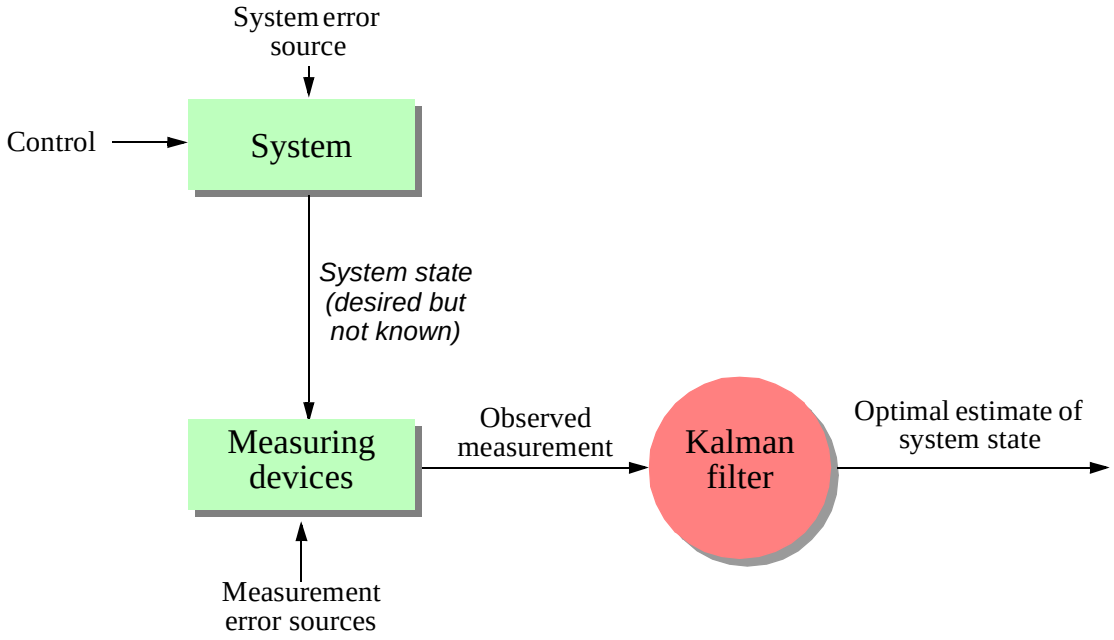


Fig 5.26 Typical Kalman filter application [46]

5.6.3 Kalman Filter Localization

The Markov localization model can represent any probability density function over robot position. This approach is very general but, due to its generality, inefficient. A successful alternative is to use a more compact representation of a specific class of probability densities. The Kalman filter does just this, and is an *optimal recursive data processing algorithm*. It incorporates all information, regardless of precision, to estimate the current value of the variable of interest. A comprehensive introduction can be found in [46] and a more detailed treatment is presented in [28].

Figure 5.26 depicts the a general scheme of Kalman filter estimation, where the system has a control signal and system error sources as inputs. A measuring device enables measuring some system states with errors. The Kalman filter is a mathematical mechanism for producing an optimal estimate of the system state based on the knowledge of the *system* and the *measuring device*, the description of the system noise and measurement errors and the uncertainty in the dynamics models. Thus the Kalman filter *fuses* sensor signals and system knowledge in an optimal way. Optimality depends on the criteria chosen to evaluate the performance and on the assumptions. Within the Kalman filter theory the system is assumed to be *linear* and *white* with *Gaussian* noise. As we have discussed earlier, the assumption of Gaussian error is invalid for our mobile robot applications but, nevertheless, the results are extremely useful. In other engineering disciplines, the Gaussian error assumption has in some cases been shown to be quite accurate [46].

We begin with a subsection that introduces Kalman filter theory, then we present an application of that theory to the problem of mobile robot localization. Finally, the third subsection will present a case study of a mobile robot that navigates indoor spaces by virtue of Kalman filter localization.

5.6.3.1 Introduction to Kalman Filter Theory

The basic Kalman filter method allows multiple measurements to be incorporated optimally into a single estimate of state. In demonstrating this, first we make the simplifying assumption that the state does not change (e.g. the robot does not move) between the acquisition of the first and second measurement. After presenting this static case, we can introduce *dynamic prediction* readily.

Static Estimation

Assume we have taken two measurements, one with an ultrasonic range sensor at time k and one with a more precise laser range sensor at time k+1. Based on each measurement we are able to estimate the robot's position. Such an estimate derived from the first sensor measurements is q_1 and the estimate of position based on the second measurement is q_2. Since we know that each measurement can be inaccurate, we wish to modulate these position estimates based on the measurement error expected from each sensor. Suppose that we use two variances σ_1^2 and σ_2^2 to predict the error associated with each measurement. We will, of course, assume a unimodal error distribution throughout the remainder of the Kalman filter approach, yielding the two robot position estimates:

$$\hat{q}_1 = q_1 \text{ with variance } \sigma_1^2 \tag{5.28}$$

$$\hat{q}_2 = q_2 \text{ with variance } \sigma_2^2. \tag{5.29}$$

So we have two measurements available to estimate the robots position. The question is, how do we *fuse* (combine) these data to get the best estimate $\hat{q}$ for the robot position?

We are assuming that there was no robot motion between time k and time k+1, and therefore we can directly apply the same weighted least square technique of Equation 4.61 in Section 4.3.1.1. Thus we write:

$$S = \sum_{i=1}^{n} w_i (\hat{q} - q_i)^2 \tag{5.30}$$

with w_i being the weight of measurement i. To find the minimum error we set the derivative of S equal to zero.

$$\frac{\partial S}{\partial \hat{q}} = \frac{\partial}{\partial \hat{q}} \sum_{i=1}^{n} w_i (\hat{q} - q_i)^2 = 2 \sum_{i=1}^{n} w_i (\hat{q} - q_i) = 0 \tag{5.31}$$

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

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

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 (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. This demonstrates that even poor measurements only increase the precision of an estimate (fig. 5.27), a result that we expect based on Information Theory.

Equation (5.35) can be rewritten as

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

or, in final form that is used in Kalman filter implementation

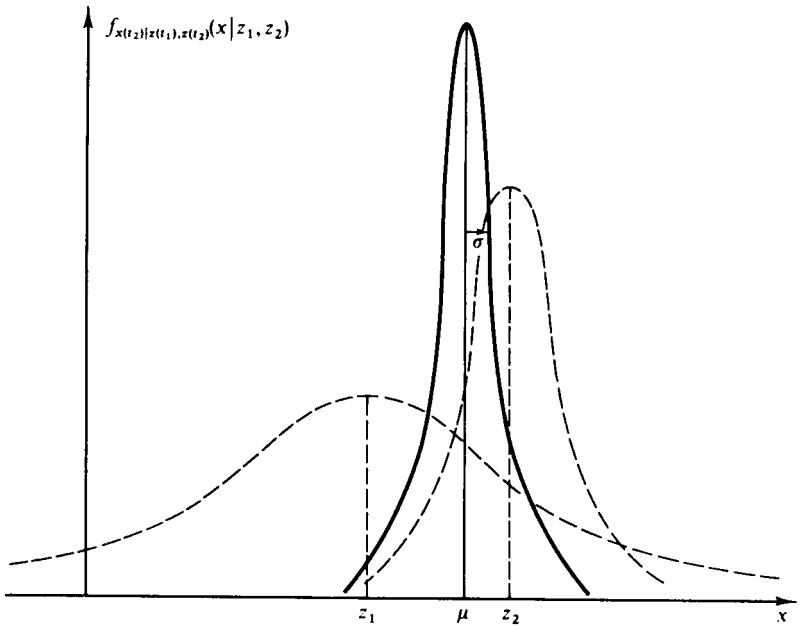


Fig 5.27 Fusing probability density of two estimates [46]

$$\hat{x}_{k+1} = \hat{x}_k + K_{k+1}(z_{k+1} - \hat{x}_k) \tag{5.38}$$

where

$$K_{k+1} = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} \quad ; \quad \sigma_k^2 = \sigma_1^2 \quad ; \quad \sigma_z^2 = \sigma_2^2 \tag{5.39}$$

Equation (5.38) tells us, that the best estimate $\hat{x}_{k+1}$ of the state x_{k+1} at time k+1 is equal to the best prediction of the value $\hat{x}_k$ before the new measurement z_{k+1} is taken, plus a correction term of an optimal weighting value times the difference between z_{k+1} and the best prediction $\hat{x}_{k+1}$ at time k+1. The updated variance of the state $\hat{x}_{k+1}$ is given using equation (5.36)

$$\sigma_{k+1}^2 = \sigma_k^2 - K_{k+1}\sigma_k^2 \tag{5.40}$$

Dynamic Estimation

Next, consider a robot that moves between successive sensor measurements. Suppose that the motion of the robot between times k and k+1 is described by the velocity u and the noise w which represents the uncertainty of the actual velocity:

$$\frac{dx}{dt} = u + w \tag{5.41}$$

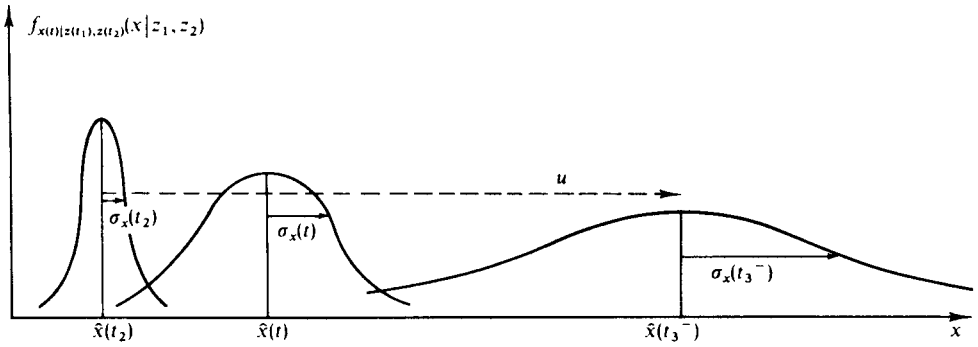


Fig 5.28 Propagation of probability density of a moving robot [46]

If we now start at time k, knowing the variance σ_k^2 of the robot position at this time and knowing the variance σ_w^2 of the motion, we obtain for the time k' just when the measurement is taken:

$$\hat{x}_{k'} = \hat{x}_k + u(t_{k+1} - t_k) \tag{5.42}$$

$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] \tag{5.43}$$

where:

$t_{k'} = t_{k+1}$

t_{k+1} and t_k are the time in seconds at k+1 and k respectively.

Thus $\hat{x}_{k'}$ is the optimal prediction of the robot's position just as the measurement is taken at time k+1. It describes the growth of position error until a new measurement is taken (fig. 5.28).

We can now rewrite equation (5.38) and (5.39) using equation (5.42) and (5.43).

$$\begin{aligned} {}_{k+1} &= \hat{x}_{k'} + K_{k+1}(z_{k+1} - \hat{x}_{k'}) \\ &= [\hat{x}_k + u(t_{k+1} - t_k)] + K_{k+1}[z_{k+1} - \hat{x}_k - u(t_{k+1} - t_k)] \end{aligned} \tag{5.44}$$

$$K_{k+1} = \frac{\sigma_{k'}^2}{\sigma_{k'}^2 + \sigma_z^2} = \frac{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]}{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] + \sigma_z^2} \tag{5.45}$$

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.

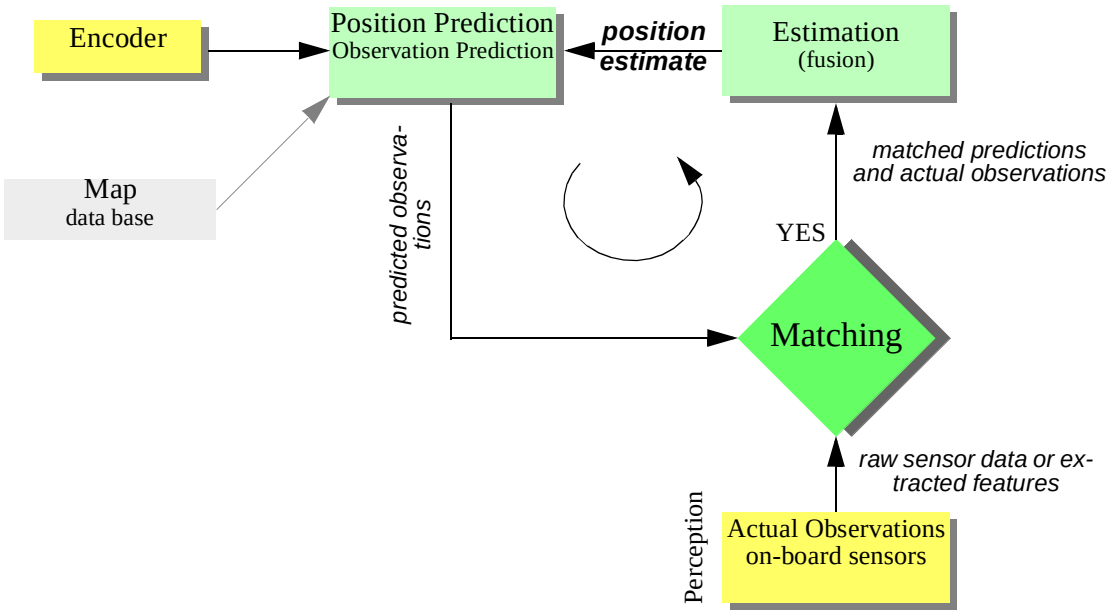


Fig 5.29 Schematic for Kalman filter mobile robot localization (see [9])

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, i.e. the robot's set of instantaneous sensor measurements, is used to update each possible robot position in the belief state individually using 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 multi-step process. The robot's total sensory input is treated, not as a monolithic whole, but as a set of extracted features that each relate to objects in the environment. Given a set of possible features, the Kalman filter is used to fuse the distance estimate from each feature to a matching object in the map. Instead of carrying out this matching process for many possible robot locations individually as in the Markov approach, the Kalman filter accomplishes the same probabilistic update by treating the whole, unimodal and Gaussian belief state at once. Figure 5.29 depicts the particular schematic for Kalman filter localization.

The first step is action update or *position prediction*, the straightforward application of a Gaussian error motion model to the robot's measured encoder travel. The robot then collects actual sensor data and extracts appropriate features (e.g. lines, doors, or even the value of a specific sensor) in the *observation* step. At the same time, based on its predicted position in the map, the robot generates a *measurement prediction* which identifies the features that the

robot expects to find and the positions of those features. In *matching* the robot identifies the best pairings between the features actually extracted during observation and the expected features due to measurement prediction. Finally, the Kalman filter can fuse the information provided by all of these matches in order to update the robot belief state in *estimation*.

In the following sub-sections these five steps are described in greater detail. The presentation is based on the work of Leonard and Durrant-Whyte [9 page 61-65].

1. Robot Position Prediction

The robot's position at time step k+1 is predicted based on its old location (at time step k) and its movement due to the control input $u(k)$:

$$\hat{p}(k+1|k) = f(\hat{p}(k|k), u(k)) \tag{5.46}$$

For the differential drive robot, $\hat{p}(k+1|k) = p'$ is derived in Equations (5.6) and (5.7) respectively.

Knowing the plant and error model, we can also compute the variance $\Sigma_p(k+1|k)$ associated with this prediction (see eq. (5.9) section 5.2.4):

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T \tag{5.47}$$

This allows us to predict the robot's position and its uncertainty after a movement specified by the control input $u(k)$. Note that the belief state is assumed to be Gaussian, and so we can characterize the believe state with just the two parameters $\hat{p}(k+1|k)$ and $\Sigma_p(k+1|k)$.

2. Observation

The second step it to obtain sensor measurements $Z(k+1)$ from the robot at time k+1. In this presentation, we assume that the observation is the result of a feature extraction process executed on the raw sensor data. Therefore, the observation consists of a set n_0 of single observations $z_j(k+1)$ extracted from various sensors. Formally each single observation can represent an extracted features such as a line or door, or even a single, raw sensor value.

The parameters of the features are usually specified in the sensor frame and therefore in a local reference frame of the robot. However, for matching we need to represent the observations and measurement predictions in the same frame $\{S\}$. In our presentation we will transform the measurement predictions from the global coordinate frame to the sensor frame $\{S\}$. This transformation is specified in the function h_i discussed in the next paragraph.

3. Measurement Prediction

We use the predicted robot position $\hat{p}(k+1|k)$ and the map $M(k)$ to generate multiple predicted feature observations z_t. Each predicted feature has its position transformed into the sensor frame:

$$\hat{z}_i(k+1) = h_i(z_t, \hat{p}(k+1|k)) \tag{5.48}$$

We can define the measurement prediction as the set containing all n_t predicted feature observations:

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

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 4.2.3 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:

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

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 hypothesis 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 [28] 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

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

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 one-dimensional 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:

$$\begin{aligned}(k+1|k+1) &= \hat{p}(k+1|k) + K(k+1) \cdot \ v(k+1) \\ &= \hat{p}(k+1|k) + K(k+1) \cdot \ [z_j(k+1) - h_i(z_t, \ \hat{p}(k+1|k))] \\ &= \hat{p}(k+1|k) + K(k+1) \cdot \ [z_j(k+1) - z_t]\end{aligned} \tag{5.57}$$

corresponding to equation (5.44).

5.6.3.3 Case study: Kalman filter localization with line feature extraction

The Pygmalion robot at EPFL is a differential drive robot that uses a laser rangefinder as its primary sensor [39], [76]. In contrast to both Dervish and Rhino, the environmental representation of Pygmalion is continuous and abstract: the map consists of a set of infinite lines describing the environment. Pygmalion's belief state is, of course, represented as a Gaussian distribution since this robot uses the Kalman filter localization algorithm. The value of its mean position μ is represented to a high level of precision, enabling Pygmalion to localize with very high precision when desired. Below, we present details for Pygmalion's implementation of the five Kalman filter localization steps. For simplicity we assume that the sensor frame $\{S\}$ is equal to the robot frame $\{R\}$. If not specified all the vectors are represented in the world coordinate system $\{W\}$.

1. Robot Position Prediction

At the time increment k the robot is at position $p(k) = \begin{bmatrix} x(k) & y(k) & \theta(k) \end{bmatrix}^T$ and its best position estimate is $\hat{p}(k|k)$. The control input $u(k)$ drives the robot to the position $p(k+1)$ (fig. 5.30).

The robot position prediction $\hat{p}(k+1)$ at the time increment k+1 can be computed from the previous estimate $\hat{p}(k|k)$ and the odometric integration of the movement. For the differential drive that Pygmalion has we can use the model (odometry) developed in section 5.2.4:

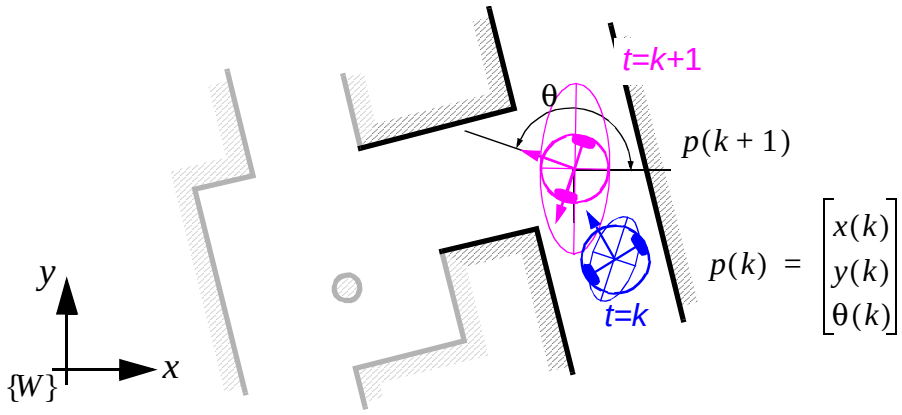


Fig 5.30 *Prediction of the robots position (magenta) based on its former position (blue) and the executed movement. The ellipses drawn around the robot positions represent the uncertainties in the x,y direction (e.g. 3σ). The uncertainty of the orientation θ is not represented in the picture.*

$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \hat{p}(k|k) + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.58}$$

with the updated covariance matrix

$$(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u \tag{5.59}$$

where

$$\Sigma_u = cov(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.60}$$

2. Observation

For line based localization, each single observation (i.e. a line feature) is extracted from the raw laser rangefinder data and consists of the two line parameters $\beta_{0,j}$, $\beta_{1,j}$ or α_j, r_j (fig. 4.36) respectively. For a rotating laser rangefinder, a representation in the polar coordinate frame is more appropriate and so we use this coordinate frame here:

$$z_j(k+1) = {}^R\begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} \tag{5.61}$$

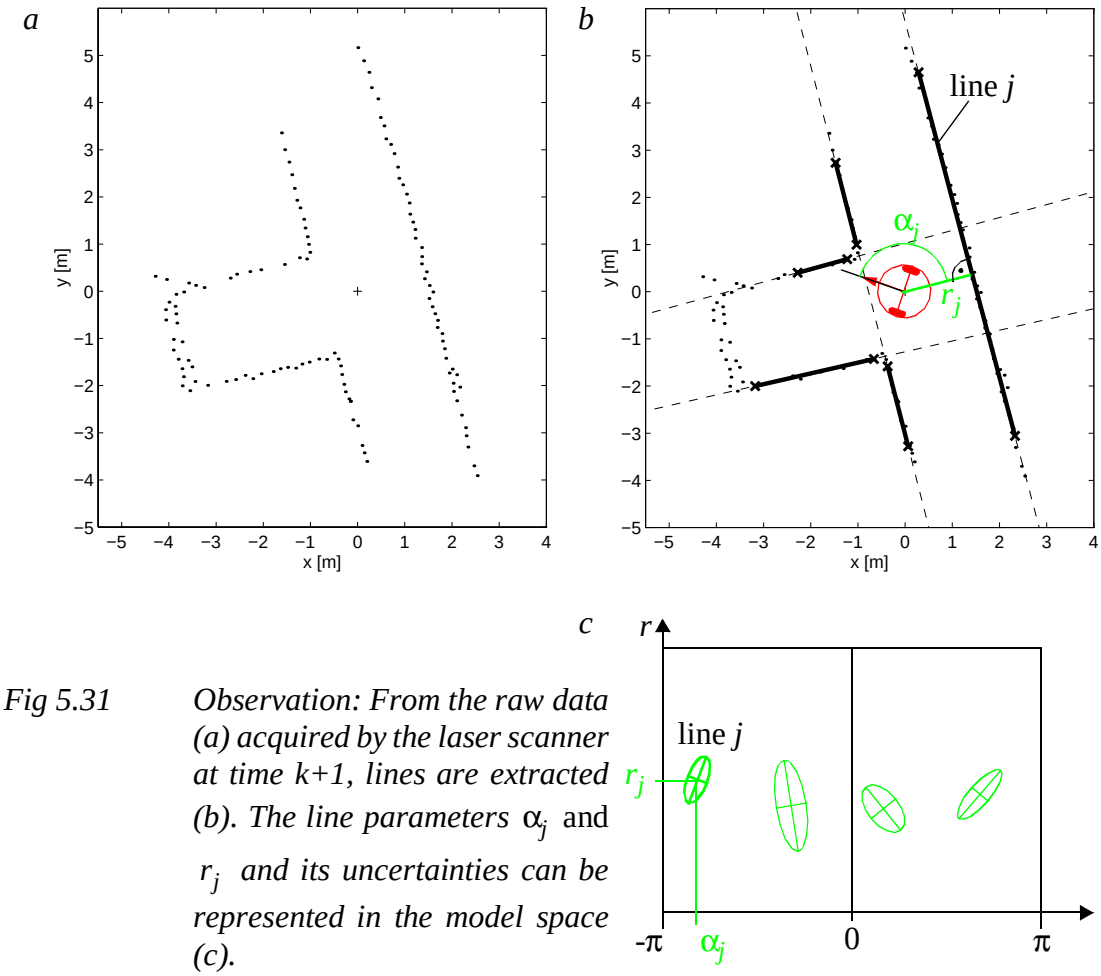


Fig 5.31 *Observation: From the raw data (a) acquired by the laser scanner at time k+1, lines are extracted (b). The line parameters α_j and r_j and its uncertainties can be represented in the model space (c).*

After acquiring the raw data at time k+1, lines and their uncertainties are extracted (fig. 5.31a/b). This leads to n_0 observed lines with $2n_0$ line parameters (5.31c) and a covariance matrix for each line that can be calculated from the uncertainties of all the measurement points contributing to each line as developed for line extraction in Section 4.3.1.1:

$$\Sigma_{R,\,j} = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix}_j \tag{5.62}$$

3. Measurement Prediction

Base on the stored map and the predicted robot position $\hat{p}(k|k)$, the measurement predictions of expected features $_{t,\,i}$ are generated (fig. 5.32). To reduce the required calculation power, there is often an additional step that first selects the possible features, in this case lines, from the whole set of features in the map. These lines are stored in the map and specified in the world coordinate system $\{W\}$. Therefore they need to be transformed to the robot frame $\{R\}$:

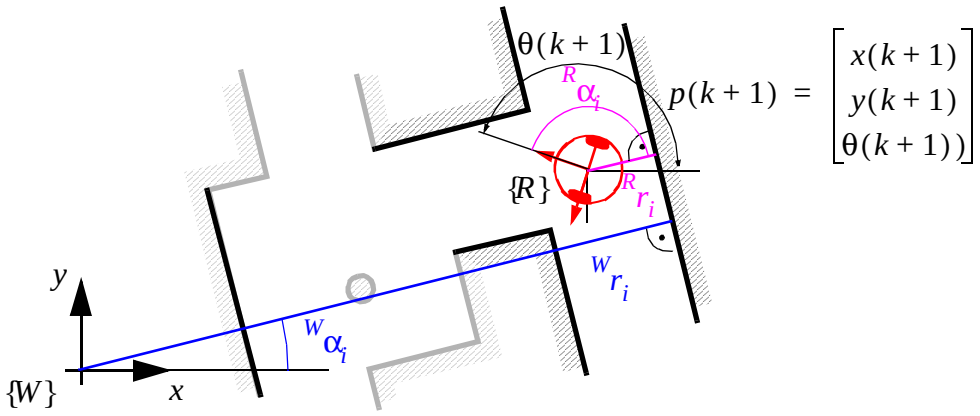


Fig 5.32 *Representation of the target position in the world coordinate frame* $\{W\}$ *and robot coordinate frame* $\{R\}$.

$$z_{t,\,i} = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{W} \rightarrow {}^{R}z_{t,\,i} = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{R} \tag{5.63}$$

According to figure (5.32), the transformation is given by

$$ {}_{i}(k+1) = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{R} = h_i(z_{t,\,i},\ \hat{p}(k+1|k)) $$

$$ = \begin{bmatrix} {}^{W}\alpha_{t,\,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,\,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,\,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,\,i})) \end{bmatrix} \tag{5.64}$$

and its Jacobian ∇h_i by

$$\nabla h_i = \begin{bmatrix} \frac{\partial \alpha_{t,\,i}}{\partial \hat{x}} & \frac{\partial \alpha_{t,\,i}}{\partial \hat{y}} & \frac{\partial \alpha_{t,\,i}}{\partial \hat{\theta}} \\ \frac{\partial r_{t,\,i}}{\partial \hat{x}} & \frac{\partial r_{t,\,i}}{\partial \hat{y}} & \frac{\partial r_{t,\,i}}{\partial \hat{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos{}^{W}\alpha_{t,\,i} & -\sin{}^{W}\alpha_{t,\,i} & 0 \end{bmatrix} \tag{5.65}$$

The measurement prediction results in predicted lines represented in the robot coordinate frame (fig. 5.33). They are uncertain, because the prediction of robot position is uncertain.

4. Matching

For matching, we must find correspondence (or a pairing) between predicted and observed features (fig. 5.34). In our case we take the Mahalanobis distance

$$ {}_{ij}^{T}(k+1) \cdot \ \Sigma_{IN,\,ij}^{-1}(k+1) \cdot \ v_{ij}(k+1) \leq g^2 \tag{5.66}$$

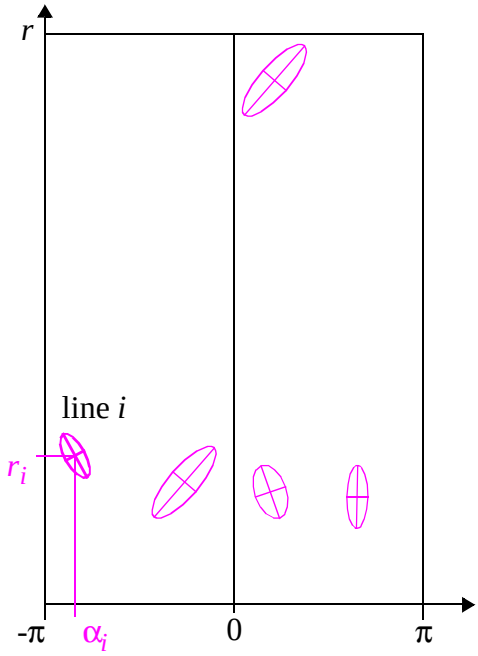


Fig 5.33 *Measurement predictions: Based on the map and the estimated robot position the targets (visible lines) are predicted. They are represented in the model space similar to the observations.*

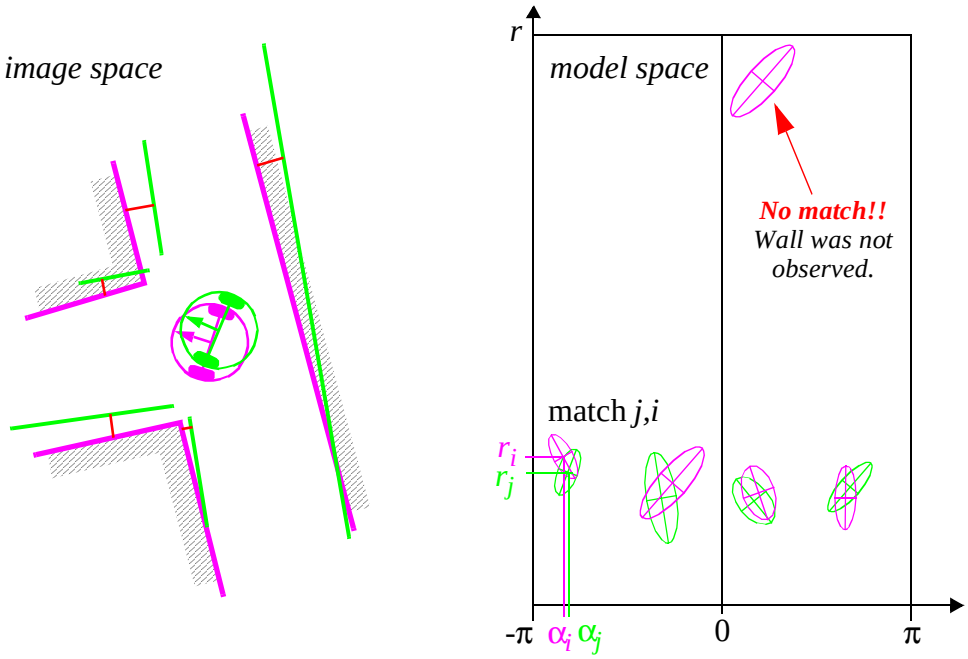


Fig 5.34 *Matching: The observations (green) and measurement prediction (magenta) are matched and the innovation and its uncertainties are calculated.*

with

$$v_{ij}(k+1) = [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))]$$
$$= \begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} - \begin{bmatrix} {}^W\alpha_{t,\,i} - {}^W\hat{\theta}(k+1|k) \\ {}^W r_{t,\,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,\,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,\,i})) \end{bmatrix} \quad (5.67)$$

$$\Sigma_{IN,\,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^T + \Sigma_{R,\,i}(k+1) \quad (5.68)$$

to enable finding the best matches while eliminating all other remaining observed and pre-

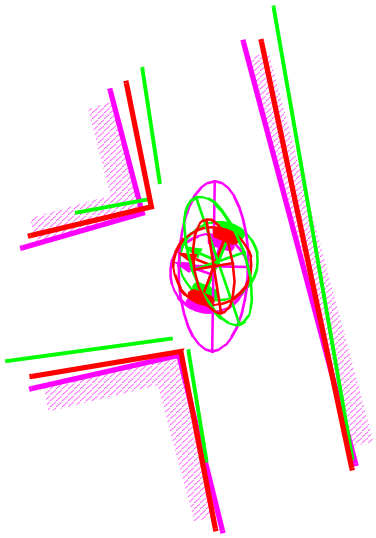


Fig 5.35 *Kalman filter estimation of the new robot position: By fusing the prediction of robot position (magenta) with the innovation gained by the measurements (green) we get the updated estimate $\hat{p}(k|k)$ of the robot position (red).*

dicted unmatched features.

5. Estimation

Applying the Kalman filter results in a final pose estimate corresponding to the weighted sum of (fig 5.35)

– *the pose estimates of each matched pairing of observed and predicted features*

– *robot position estimation based on odometry and observation positions*

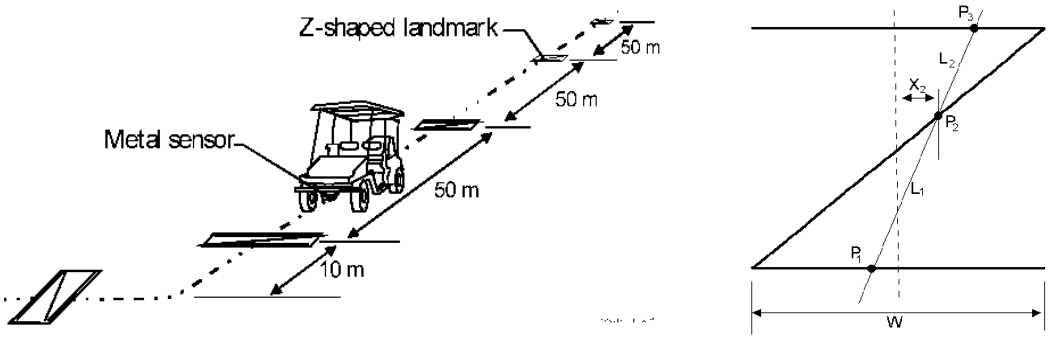


Fig 5.36 Z-shaped landmarks on the ground

5.7 Other Examples of Localization Systems

Markov localization and Kalman filter localization have been two extremely popular strategies for research mobile robot systems navigating indoor environments. They have strong formal bases and therefore well-defined behavior. But there are a large number of other localization techniques that have been used with varying degrees of success on commercial and research mobile robot platforms. We will not explore the space of all localization systems in detail. Refer to surveys such as [4] for such information.

There are, however, several categories of localization techniques that deserve mention. Not surprisingly, many implementations of these techniques in commercial robotics employ modifications of the robot's environment, something that the Markov localization and Kalman filter localization communities eschew. In the following sections, we briefly identify the general strategy incorporated by each category and reference example systems, including as appropriate those that modify the environment and those that function without environmental modification.

Landmark-based navigation

Landmarks are generally defined as passive objects in the environment that provide a high degree of localization accuracy when they are within the robot's field of view. Mobile robots that make use of landmarks for localization generally use artificial markers that have been placed by the robot's designers to make localization easy.

The control system for a landmark-based navigator consists of two discrete phases. When a landmark is in view, the robot localizes frequently and accurately, using action update and perception update to track its position without cumulative error. But when the robot is in no landmark "zone," then only action update occurs, and the robot accumulates position uncertainty until the next landmark enters the robot's field of view.

The robot is thus effectively *dead-reckoning* from landmark zone to landmark zone. This in turn means the robot must consult its map carefully, ensuring that each motion between landmarks is sufficiently short, given its motion model, that it will be able to localize successful upon reaching the next landmark.

Figure 5.36 shows one instantiation of landmark-based localization. The particular shape of the landmarks enables reliable and accurate pose estimation by the robot, which must travel

using *dead reckoning* between the landmarks.

One key advantage of the landmark-based navigation approach is that a strong formal theory has been developed for this general system architecture [113]. In this work, the authors have shown precise assumptions and conditions which, when satisfied, guarantee 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 meters 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 is usually required to cover a large factory area or research laboratory. For example, the Robotics Laboratory at Stanford made use of approximately 30 discrete landmarks, all affixed individually to the ceiling.

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 towards globally unique localization.

Figure (4.50) 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 for example Figure 4.51 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 5.37 is another example in which the robot's sensor values are transformed into an identifier of location. However, due to the limited infor-

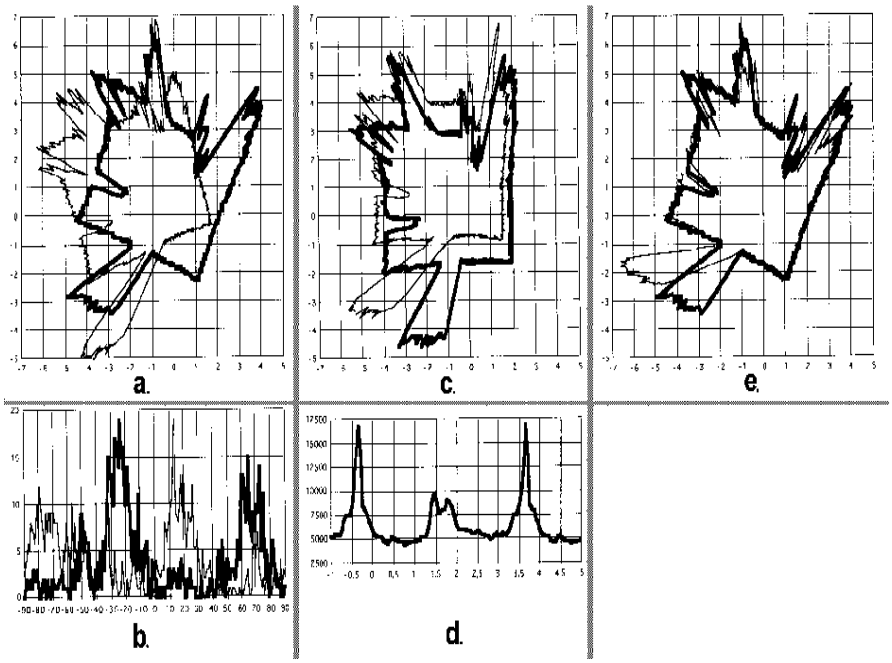


Fig 5.37 The angular histogram: Example

mation content of sonar ranging strikes, it is likely that two *places* in the robot's environment may have angular histograms that are too similar to be differentiated successfully.

One way of attempting to gather sufficient sonar information for global localization is to allow the robot time to gather a large amount of sonar data into a local evidence grid (i.e. occupancy grid) first, then match the local evidence grid with a global metric map of the environment. In [115] the researchers demonstrate such a system as able to localize on-the-fly even as significant changes are made to the environment, degrading the fidelity of the map. Most interesting is that the local evidence grid represents information well enough that it can be used to correct and update the map over time, thereby leading to a localization system that provides corrective feedback to the environment representation directly. This is similar in spirit to the idea of taking rejected observed features in the Kalman filter localization algorithm and using them to create new features in the map.

A most promising, new method for globally unique localization is called *Mosaic-based localization* [114]. This fascinating approach takes advantage of an environmental feature that is rarely used by mobile robots: fine-grained floor texture. This method succeeds primarily because of the recent ubiquity of very fast processors, very fast cameras and very large storage media.

The robot is fitted with a high-quality high-speed CCD camera pointed toward the floor, ideally situated between the robot's wheels and illuminated by a specialized light pattern off the camera axis to enhance floor texture. The robot begins by collecting images of the entire floor in the robot's workspace using this camera. Of course the memory requirements are significant, requiring a 10GB drive in order to store the complete image library of a 300 x 300 meter area.

Once the complete image mosaic is stored, the robot can travel any trajectory on the floor while tracking its own position without difficulty. Localization is performed by simply re-

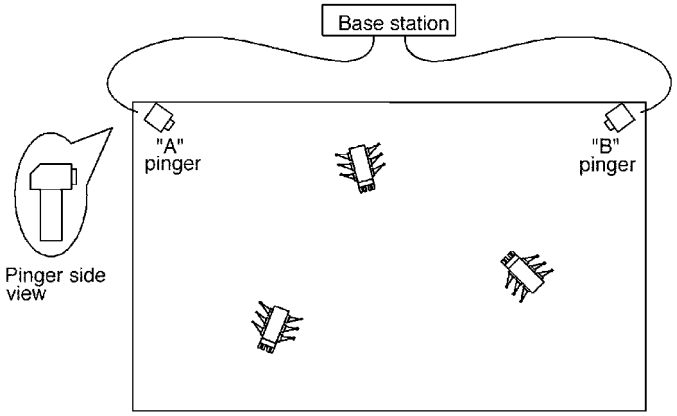


Fig 5.38 Active ultrasonic beacons.

cording one image, performing action update, then performing perception update by matching the image to the mosaic database using simple techniques based on image database matching. The resulting performance has been impressive: such a robot has been shown to localize repeatedly with 1mm precision while moving at 25 km/hr.

The key advantage of globally unique localization is that, when these systems function correctly, they greatly simplify robot navigation. The robot can move to any point and will always be assured of localizing by collecting a sensor scan.

But the main disadvantage of globally unique localization is that it is likely that this method will *never* offer a complete solution to the localization problem. There will always be cases where local sensory information is truly ambiguous and, therefore, globally unique localization using only current sensor information is unlikely to succeed. Humans often have excellent *local positioning systems*, particularly in non-repeating and well-known environments such as their homes. However, there are a number of environments in which such immediate localization is challenging even for humans: consider hedge mazes and large new office buildings with repeating halls that are identical. Indeed, the mosaic-based localization prototype described above encountered such a problem in its first implementation. The floor of the factory floor had been freshly painted and was thus devoid of sufficient micro-fractures to generate texture for correlation. Their solution was to modify the environment after all, painting random texture onto the factory floor.

Positioning Beacon systems

One of the most reliable solutions to the localization problem is to design and deploy an active beacon system specifically for the target environment. This is the preferred technique used by both industry and military applications as a way of ensuring the highest possible reliablility of localization. The GPS system can be considered as just such a system (see Section 4.1.5.1).

Figure 5.38 depicts one such beacon arrangement for a collection of robots. Just as with GPS, by designing a system whereby the robots localize passively while the beacons are active, any number of robots can simultaneously take advantage of a single beacon system. As

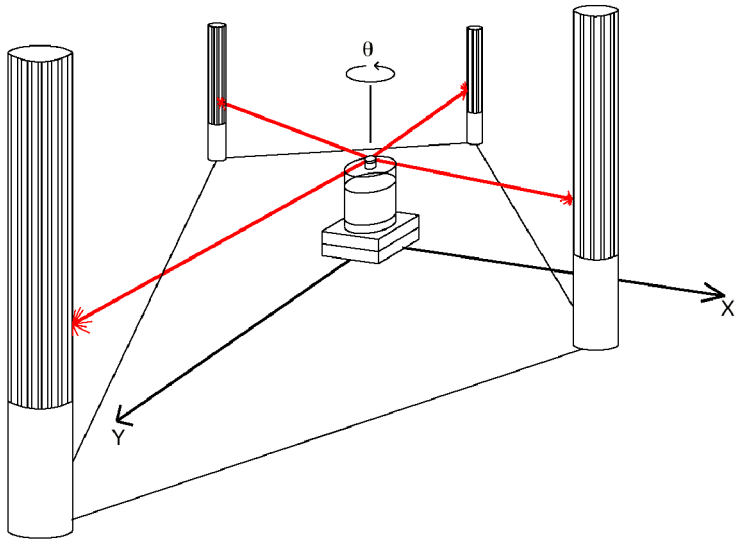


Fig 5.39 Passive optical beacons

with most beacon systems, the design depicted depends foremost upon geometric principles to effect localization. In this case the robots must know the positions of the two pinger units in the global coordinate frame in order to localize themselves to the global coordinate frame.

A popular type of beacon system in industrial robotic applications is depicted in Figure 5.39. In this case beacons are retroreflective markers that can be easily detected by a mobile robot based on their reflection of energy back to the robot. Given known positions for the optical retroreflectors, a mobile robot can identify its position whenever it has three such beacons in sight simultaneously. Of course, a robot with encoders can localize over time as well, and does not need to measure its angle to all three beacons at the same instant.

The advantage of such beacon-based systems is usually extremely high engineered reliability. By the same token, significant engineering usually surrounds the installation of such a system in a specific commercial setting. Therefore, moving the robot to a different factory floor will be both time-consuming and expensive. Usually, even changing the routes used by the robot will require serious re-engineering.

Route-based localization

Even more reliable than beacon-based systems are route-based localization strategies. In this case, the route of the robot is explicitly marked so that it can determine its position, not relative to some global coordinate frame, but relative to the specific path it is allowed to travel. There are many techniques for marking such a route and the subsequent intersections. In all cases, one is effectively creating a railway system, except that the railway system is somewhat more flexible and certainly more human-friendly than a physical rail. For example, high uv-reflective, optically transparent paint can mark the route such that only the robot, using a specialized sensor, easily detects it. Alternatively, a guide wire buried underneath the hall can be detected using inductive coils located on the robot chassis.

In all such cases, the robot localization problem is effectively trivialized by forcing the robot to always follow a prescribed path. To be fair, there are new industrial *unmanned guided*

vehicles that do deviate briefly from their route in order to avoid obstacles. Nevertheless, the cost of this extreme reliability is obvious: the robot is much more inflexible give such localization means, and therefore any change to the robot's behavior requires significant engineering and time.

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 is not so, 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 environment map. Of course a mobile robot's sensors have only 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 a 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 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 sub-section [75].

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 5.40 shows a general schematic incorporating map building and maintenance into the standard localization loop depicted by Figure (5.29) during discussion of Kalman filter localization [9]. The added arcs represent the additional flow of information that occurs when there is an imperfect match between observations and measurement predictions.

Unexpected observations will affect the creation of new features in the map whereas unobserved measurement predictions will affect the removal of features from the map. As discussed earlier, each specific prediction or observation has an unknown exact value and so it

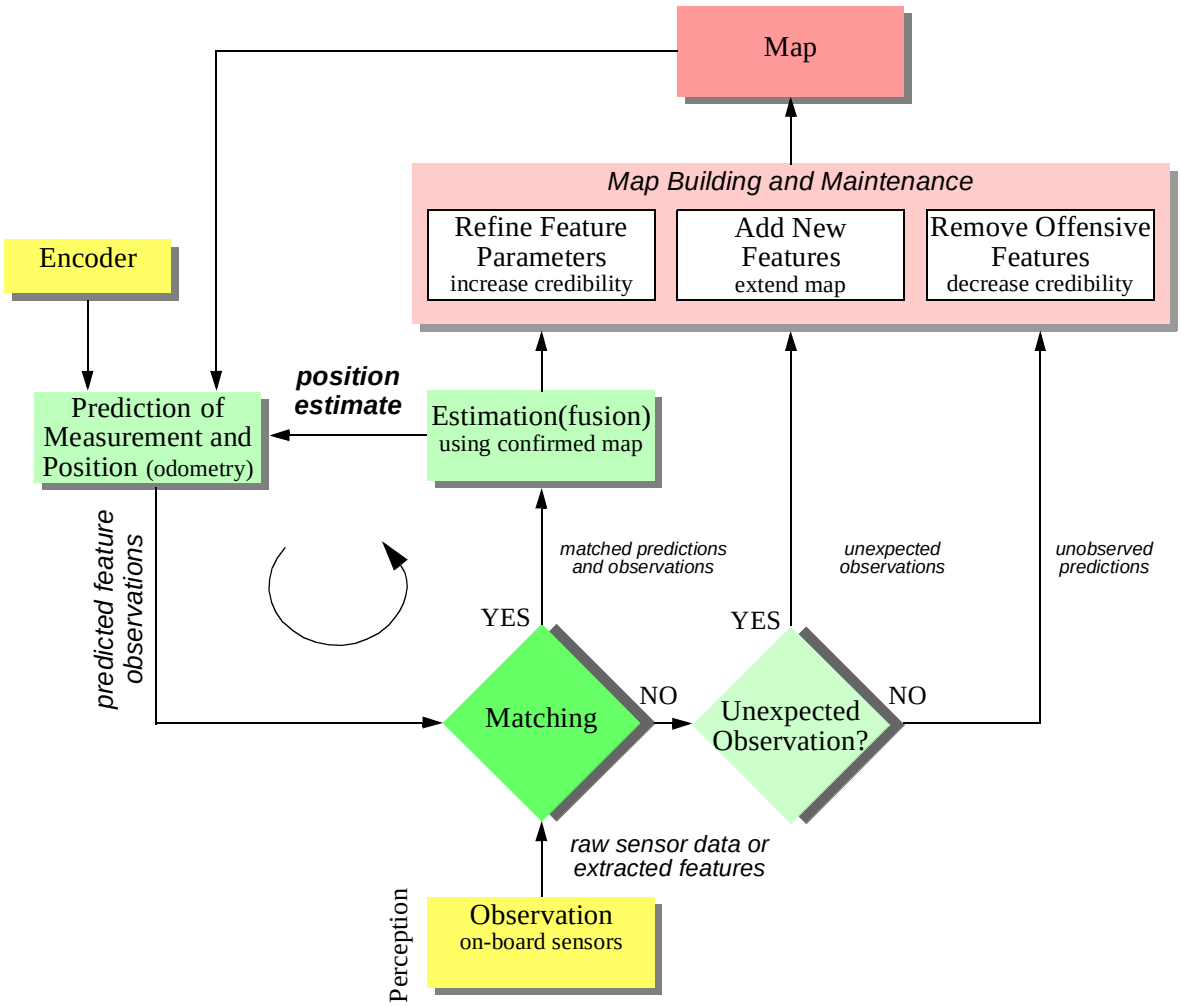


Fig 5.40 General schematic for concurrent localization and map building (see [9])

is represented by a distribution. The uncertainties of all of these quantities must be considered throughout this process.

The new type of map we are creating not only has features in it as did previous maps, but it also has varying degrees of probability that each feature is indeed part of the environment.

We represent this new map M with a set n of probabilistic feature locations $\hat{z}_t$, each with the covariance matrix Σ_t and an associated *credibility factor* c_t between 0 and 1 quantifying the belief in the existence of the feature in the environment (see Fig. (5.41)):

$$M = \{\hat{z}_t,\ \Sigma_t,\ c_t \mid (1 \le t \le n)\} \tag{5.69}$$

In contrast to the map used for Kalman filter localization previously, the map M is not assumed to be precisely known because it will be created by an uncertain robot over time. This is why the features $\hat{z}_t$ are described with associated covariance matrices Σ_t.

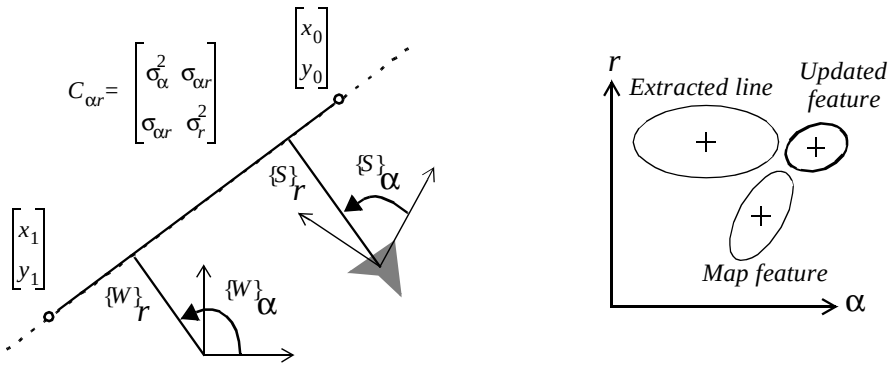


Fig 5.41 Uncertainties in the map.

Just as with Kalman filter localization, the matching step yields has three outcomes in regard to measurement predictions and observations: *matched prediction and observations*, *unexpected observations* and *unobserved predictions*. Localization, or the position update of the robot, proceeds as before. However, the map is also updated now, using all three outcomes and complete propagation of all the correllated uncertainties (see [9] for more details).

An interesting variable is the credibility factor c_t, which governs the likelihood that the mapped feature is indeed in the environment. How should the robot's failure to match observed features to a particular map feature reduce that map feature's credibility? And also, how should the robot's success at matching a mapped feature increase the chance that the mapped feature is "correct?" In [9] the following function is proposed for calculating credibility:

$$c_t(k) = 1 - e^{-\left(\frac{n_s}{a} - \frac{n_u}{b}\right)} \tag{5.70}$$

where a and b define the learning and forgetting rate and n_s and n_u are the number of matched and unobserved predictions up to time k, respectively. The update of the covariance matrix Σ_t can be done similarly to the position update seen in previous section. In map-building the feature positions and the robot's position are strongly correlated. This forces us to use a *stochastic map*, in which all cross-correlations must be updated in each cycle [73, 74, 75].

The stochastic map consists of a stacked system state vector:

$$X = \left[x_r(k)\ x_1(k)\ x_2(k)\ \ldots x_n(k)\right]^T \tag{5.71}$$

and a system state covariance matrix:

$$\Sigma = \begin{bmatrix} C_{rr} & C_{r1} & C_{r2} & \dots C_{rn} \\ C_{1r} & C_{11} & \dots & \dots C_{1n} \\ C_{2r} & \dots & \dots & \dots C_{2n} \\ \dots & \dots & \dots & \dots \dots \\ C_{nr} & C_{n1} & C_{n2} & \dots C_{nn} \end{bmatrix} \quad (5.72)$$

where the index r stands for the robot and the index $i = 1$ to n for the features in the map.

In contrast to localization based on an *a priori* accurate map, in the case of a stochastic map the cross-correlations must be maintained and updated as the robot is performing automatic map-building. During each localization cycle, the cross-correlations robot-to-feature and feature-to-robot are also updated. In short, this optimal approach requires every value in the map to depend on every other value, and therein lies the reason that such a complete solution to the automatic mapping problem is beyond the reach of even today's computational resources.

5.8.2 Other Mapping techniques

The mobile robotics research community has spent significant research effort on the problem of automatic mapping, and has demonstrating working systems in many environments without having solved the complete stochastic map problem described earlier. This field of mobile robotics research is extremely large, and this text will not present a comprehensive survey of the field. Instead, we present below two key considerations associated with automatic mapping, together with brief discussions of the approaches taken by several automatic mapping solutions to overcome these challenges.

5.8.2.1 Cyclic environments

Possibly the single hardest challenge for automatic mapping to be conquered is to correctly map cyclic environments. The problem is simple: given an environment that has one or more loops or cycles (e.g. four hallways that intersect to form a rectangle), create a globally consistent map for the whole environment.

This problem is hard because of the fundamental behavior of automatic mapping systems: the maps they create are not perfect. And, given any local imperfection, accumulating such imperfections over time can lead to arbitrarily large *global* errors between a map, at the macro level, and the real world, as shown in Figure (5.42). Such global error is usually irrelevant for mobile robot localization and navigation. After all, a warped map will still serve the robot perfectly well so long as the local error is bounded. However, an extremely large loop still eventually returns to the same spot, and the robot must be able to note this fact in its map. Therefore, global error does indeed matter in the case of cycles.

In some of the earliest work attempting to solve the cyclic environment problem, [116] used a purely topological representation of the environment, reasoning that the topological representation only captures the most abstract, most important features and avoids a great deal of

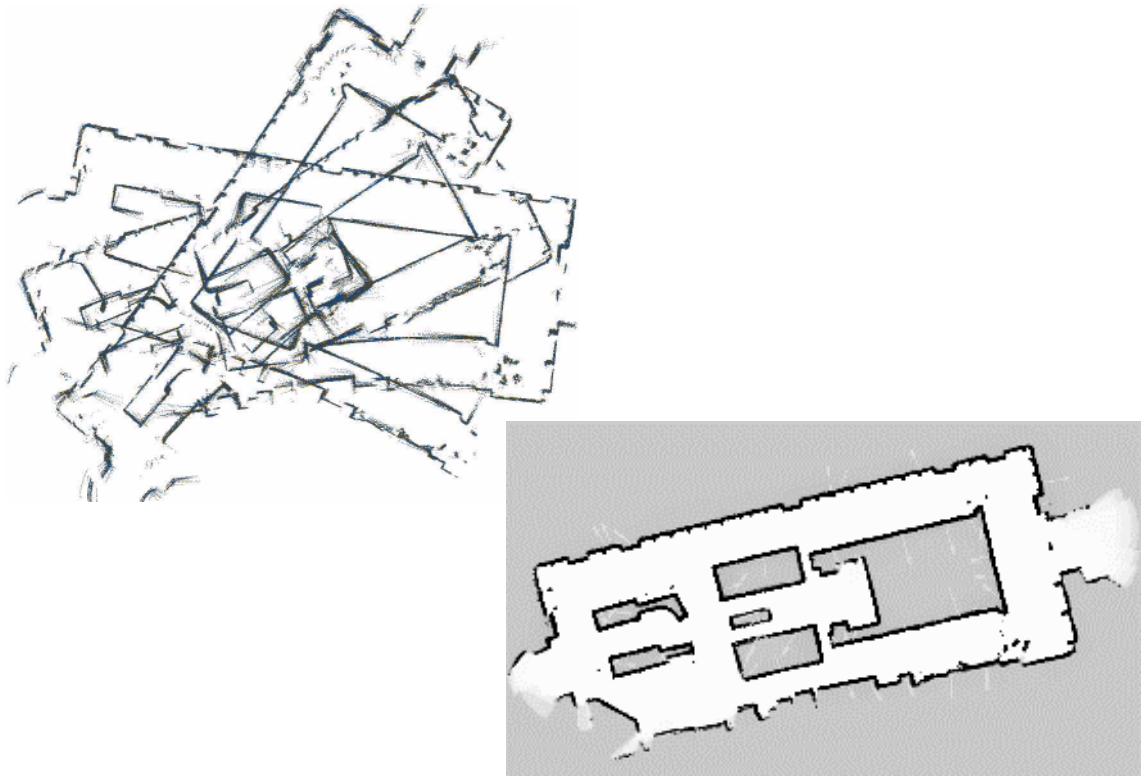


Fig 5.42 *Cyclic Environments: A naive, local mapping strategy with small local error leads to global maps that have a significant error, as demonstrated by this real-world run on the left. By applying topological correction, the grid map on the right is extracted [47].*

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 *sub-maps* [117, 118, 119]. Each *sub-map* is treated as a single sensor during the robot's position update. The advantage of this approach is two-fold. Because odometry is relatively accurate over small distances, the relative registration of features and raw sensor strikes in a local *sub-map* 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 *sub-maps*, explicitly identifying the

loops first at the topological level. In the case of [118] for example, the topological level loop is identified by a human who pushes a button at a known landmark position. In the case of [119] the topological level loop is determined by performing correspondence tests between *sub-maps*, postulating that two *sub-maps* 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 automonous mapping solutions but even 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 3:00 past midnight. However, in a great many cases this assumption is incorrect. In the case of wide-open spaces that are popular gathering places for humans, there is rapid change in the freespace and a vast majority of sensor strikes represent detection of the transient humans rather than fixed surfaces such as the perimeter wall. Another class of dynamic environments are spaces such as factory floors and warehouses, where the objects being stored redefine the topology of the pathways on a day-to-day basis as shipments are moved in and out.

In all such dynamic environments, an automatic mapping system should capture the *salient* objects detected by its sensors and, furthermore, the robot should have the flexibility to modify its map as the position of these salient objects changes. The subject of *continuous mapping*, or mapping of dynamic environments is to some degree a direct outgrowth of successful strategies for automatic mapping of unfamiliar environments. For example, in the case of stochastic mapping using the credibility factor c_t mechanism, the credibility equation can continue to provide feedback regarding the probability of existence of various mapped features after the initial map creation process is ostensibly complete. Thus, a mapping system can become a map-modifying system by simply continuing to operate. This is most effective, of course, if the mapping system is real-time and incremental. If map construction requires off-line global optimization, then the desire to make small-grained, incremental adjustments to the map is more difficult to satisfy.

Earlier we stated that a mapping system should capture only the *salient* objects detected by its sensors. One common argument for handling the detection of, for instance, humans in the environment is that mechanisms such as c_t can take care of all features that did not deserve to be mapped in the first place. For example, in [117] the authors develop a system based on a set of local occupany grids (called *evidence grids*) and a global occupancy grid.

Each time the robot's most recent local evidence grid is used to update a region of the global occupancy grid, extraneous occupied cells in the global occupancy grid are freed if the local occupancy grid detected no objects (with high confidence) at those same positions.

The general solution to the problem of detecting salient features, however, begs a solution to the perception problem in general. When a robot's sensor system can reliably detect the difference between a wall and a human, using for example a vision system, then the problem of mapping in dynamic environments will become significantly more straightforward.

We have discussed just two important considerations for automatic mapping. There is still a great deal of research activity focusing on the general map building and localization problem [9, 6, 47, 48, 49, 50, 75, 77]. However, there are few groups working on the general problem of probabilistic map building (i.e. stochastic maps) and, so far, a consistent and absolutely general solution has yet to be found. This field is certain to produce significant new results in the next several years, and as the perceptual power of robots improves we expect the payoff to be greatest here.

5 Mobile Robot Localization

5.1 Introduction

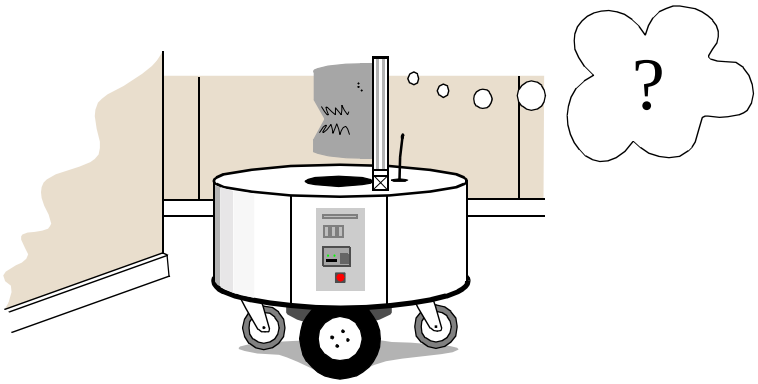


Fig 5.1 *Where am I?*

Navigation is one of the most challenging competencies required of a mobile robot. Success in navigation requires success at the four building blocks of navigation (fig. 5.2): *perception*- the robot must interpret its sensors to extract meaningful data; *localization*- the robot must determine its position in the environment; *cognition*- the robot must decide how to act to achieve its goals; and *motion control*- the robot must modulate its motor outputs to achieve the desired trajectory.

Of these four components, localization has received the greatest research attention in the past decade and, as a result, significant advances have been made on this front. In this chapter, we will explore the successful localization methodologies of recent years. First, Section 5.2 describes how sensor and effector uncertainty is responsible for the difficulties of localization. Then, Section 5.3 describes two extreme approaches to dealing with the challenge of robot localization: avoiding localization altogether, and performing explicit map-based localization. The remainder of the chapter discusses the question of representation, then presents case studies of successful localization systems using a variety of representations and techniques to achieve mobile robot localization competence.

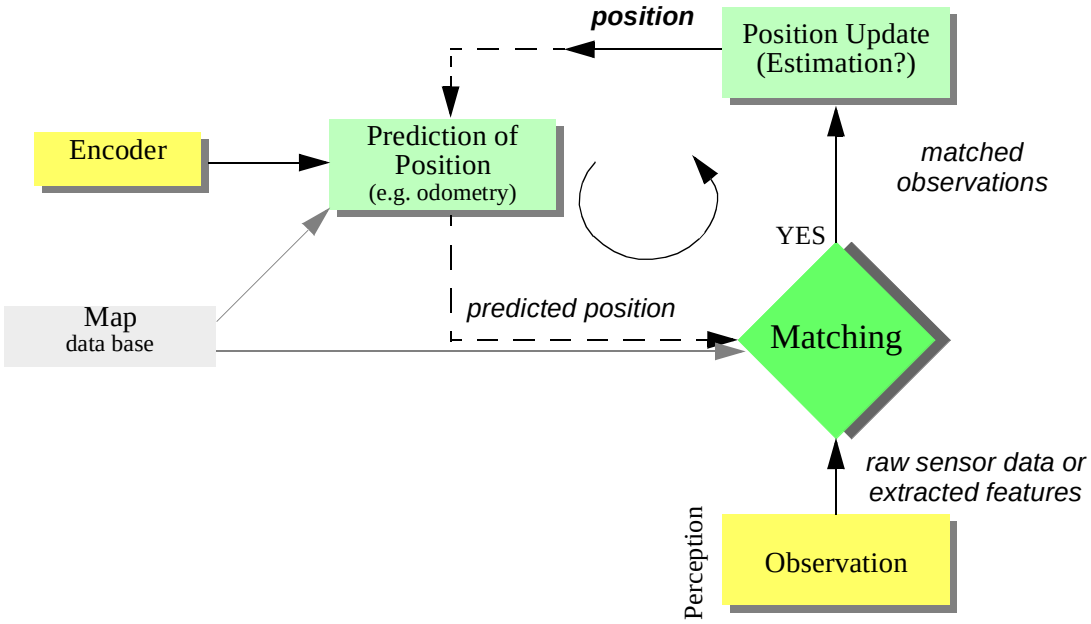


Fig 5.2 *General schematic for mobile robot localization.*

5.2 The Challenge of Localization: noise and aliasing

If one could attach an accurate *GPS* (Global Position System) sensor to a mobile robot, much of the localization problem would be obviated. The *GPS* would inform the robot of its exact position and orientation, indoors and outdoors, so that the answer to the question, "Where am I?" would always be immediately available. Unfortunately, such a sensor is not currently practical. The existing *GPS* network provides accuracy to within several meters, which is unacceptable for localizing human-scale mobile robots as well as miniature mobile robots such as desk robots and the body-navigating nano-robots of the future. Furthermore, *GPS* technologies cannot function indoors or in obstructed areas and are thus limited in their workspace.

But, looking beyond the limitations of *GPS*, localization implies more than knowing one's absolute position in the Earth's reference frame. Consider a robot that is interacting with humans. This robot may need to identify its absolute position, but its relative position with respect to target humans is equally important. Its localization task can include identifying humans using its sensor array, then computing its relative position to the humans. Furthermore, during the *Cognition* step a robot will select a strategy for achieving its goals. If it intends to reach a particular location, then localization may not be enough. The robot may need to acquire or build an environmental model, a *map*, that aids it in planning a path to the goal. Once again, localization means more than simply determining an absolute pose in space; it means building a map, then identifying the robot's position relative to that map.

Clearly, the robot's sensors and effectors play an integral role in all the above forms of localization. It is because of the inaccuracy and incompleteness of these sensors and effectors that localization poses difficult challenges. This section identifies important aspects of this sensor and effector suboptimality.

5.2.1 Sensor Noise

Sensors are the fundamental robot input for the process of *perception*, and therefore the degree to which sensors can discriminate world state is critical. *Sensor noise* induces a limitation on the consistency of sensor readings in the same environmental state and, therefore, on the number of useful bits available from each sensor reading. Often, the source of sensor noise problems is that some environmental features are not captured by the robot's representation and are thus overlooked.

For example, a vision system used for indoor navigation in an office building may use the color values detected by its color CCD camera. When the sun is hidden by clouds, the illumination of the building's interior changes due to windows throughout the building. As a result, hue values are not constant. The color CCD appears noisy from the robot's perspective as if subject to random error, and the hue values obtained from the CCD camera will be unusable, unless the robot is able to note the position of the Sun and clouds in its representation.

Illumination dependency is only one example of the apparent noise in a vision-based sensor system. Picture jitter, signal gain, blooming and blurring are all additional sources of noise, potentially reducing the useful content of a color video image.

Consider the noise level (i.e. apparent random error) of ultrasonic range-measuring sensors (e.g. sonars) as we discussed in Section 4.1.2.3. When a sonar transducer emits sound towards a relatively smooth and angled surface, much of the signal will coherently reflect away, failing to generate a return echo. Depending on the material characteristics, a small amount of energy may return nonetheless. When this level is close to the gain threshold of the sonar sensor, then the sonar will, at times, succeed and, at other times, fail to detect the object. From the robot's perspective, a virtually unchanged environmental state will result in two different possible sonar readings: one short, and one long.

The poor signal to noise ratio of a sonar sensor is further confounded by interference between multiple sonar emitters. Often, research robots have between 12 to 48 sonars on a single platform. In acoustically reflective environments, multipath interference is possible between the sonar emissions of one tranducer and the echo detection circuitry of another transducer. The result can be dramatically large errors (i.e. underestimation) in ranging values due to a set of coincidental angles. Such errors occur rarely, less than 1% of the time, and are virtually random from the robot's perspective.

In conclusion, sensor noise reduces the useful information content of sensor readings. Clearly, the solution is to take multiple readings into account, employing temporal fusion or multi-sensor fusion to increase the overall information content of the robot's inputs.

5.2.2 Sensor Aliasing

A second shortcoming of mobile robot sensors causes them to yield little information content, further exacerbating the problem of perception and, thus, localization. The problem, known as *sensor aliasing*, is a phenomenon that humans rarely encounter. The human sensory system, particularly the visual system, tends to receive unique inputs in each unique local state. In other words, every different place looks different. The power of this unique mapping is only apparent when one considers situations where this fails to hold. Consider moving through an unfamiliar building that is completely dark. When the visual system sees only black, one's localization system quickly degrades. Another useful example is that of a human-sized maze made from tall hedges. Such mazes have been created for centuries, and humans find them extremely difficult to solve without landmarks or clues because, without visual uniqueness, human localization competence degrades rapidly.

In robots, the non-uniqueness of sensors readings, or *sensor aliasing*, is the norm and not the exception. Consider a narrow-beam rangefinder such as ultrasonic or infrared rangefinders. This sensor provides range information in a single direction without any additional data regarding material composition such as color, texture and hardness. Even for a robot with several such sensors in an array, there are a variety of environmental states that would trigger the same sensor values across the array. Formally, there is a many-to-one mapping from environmental states to the robot's perceptual inputs. Thus, the robot's percepts cannot distinguish from among these many states. A classical problem with sonar-based robots involves distinguishing between humans and inanimate objects in an indoor setting. When facing an apparent obstacle in front of itself, should the robot say "Excuse me" because the obstacle may be a moving human, or should the robot plan a path around the object because it may be a cardboard box? With sonar alone, these states are aliased and differentiation is impos-

sible.

The problem posed to navigation because of sensor aliasing is that, even with noise-free sensors, the amount of information is generally insufficient to identify the robot's position from a single percept reading. Thus techniques must be employed by the robot programmer that base the robot's localization on a series of readings and, thus, sufficient information to recover the robot's position over time.

5.2.3 Effector Noise

The challenges of localization do not lie with sensor technologies alone. Just as robot sensors are noisy, limiting the information content of the signal, so robot effectors are also noisy. In particular, a single action taken by a mobile robot may have several different possible results, even though from the robot's point of view the initial state before the action was taken is well-known.

In short, mobile robot effectors introduce uncertainty about future state. Therefore the simple act of moving tends to increase the uncertainty of a mobile robot. There are, of course, exceptions. Using *cognition*, the motion can be carefully planned so as to minimize this effect, and indeed sometimes to actually result in more certainty. Furthermore, when the robot actions are taken in concert with careful interpretation of sensory feedback, it can compensate for the uncertainty introduced by noisy actions using the information provided by the sensors.

First, however, it is important to understand the precise nature of the effector noise that impacts mobile robots. It is important to note that, from the robot's point of view, this error in motion is viewed as error in odometry, or the robot's inability to estimate its own position over time using knowledge of its kinematics and dynamics. The true source of error generally lies in an incomplete model of the environment. For instance, the robot does not model the fact that the floor may be sloped, the wheels may slip, and a human may push the robot. All of these un-modeled sources of error result in inaccuracy between the physical motion of the robot, the intended motion of the robot and the proprioceptive sensor estimates of motion.

In odometry (wheel sensors only) and dead reckoning (also heading sensors) the position update is based on *proprioceptive* sensors. The movement of the robot, sensed with wheel encoders and /or heading sensors is integrated to compute position. Because the sensor measurement errors are integrated, the position error accumulates over time. Thus the position has to be updated from time to time by other localization mechanisms. Otherwise the robot is not able to maintain a meaningful position estimate in long run.

In the following we will concentrate on odometry based on the wheel sensor readings of a differential drive robot only (see also [3, 40, 41]). Using additional heading sensors (e.g. gyroscope) can help to reduce the cumulative errors, but the main problems remain the same.

There are many sources of odometric error, from environmental factors to resolution:

- Limited resolution during integration (time increments, measurement resolution, etc.)
- Misalignment of the wheels (deterministic)

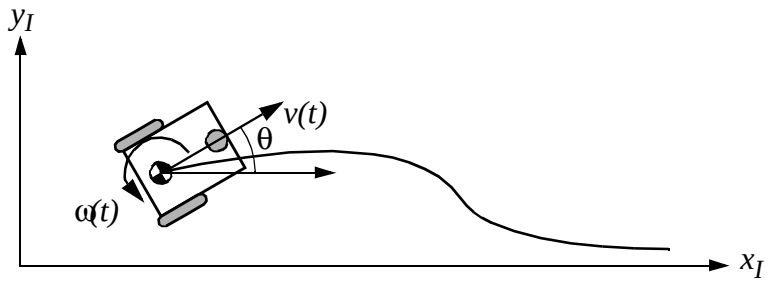


Fig 5.3 Movement of a differential drive robot

- Unequal wheel diameter (deterministic)
- Variation in the contact point of the wheel
- Unequal floor contact (slipping, non-planar 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 ***non-deterministic*** *(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:

- Range error: integrated path length (distance) of the robots movement
  -> sum of the wheel movements
- Turn error: similar to range error, but for turns
  -> difference of the wheel motions
- 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 contribute 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:

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

$$\Delta y = \Delta s \sin(\theta + \Delta\theta / 2) \tag{5.3}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{5.5}$$

where:

$(\Delta x;\Delta y;\Delta\theta)$: Path traveled in the last sampling interval

$\Delta s_r;\Delta s_l$: Traveled distances for right and left wheel respectively

b: Distance between the two wheels of differential drive robot

Thus we get the updated position p':

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta / 2) \\ \Delta s \sin(\theta + \Delta\theta / 2) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta / 2) \\ \Delta s \sin(\theta + \Delta\theta / 2) \\ \Delta\theta \end{bmatrix} \tag{5.6}$$

By using the relation for $(\Delta s;\Delta\theta)$ of equations (5.4) and (5.5) we further obtain the basic equation for odometric position update (for differential drive robots):

$$p' = f(x,\ y,\ \theta,\ \Delta s_r,\ \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.7}$$

As we discussed earlier, odometric position updates can give only a very rough estimate of the actual position. Due to integration errors of the uncertainties of p and the motion errors during the incremental motion $(\Delta s_r;\Delta s_l)$ the position error based on odometry integration grows with time.

In the next step we will establish an error model for the integrated position p' to obtain the covariance matrix $\Sigma_{p'}$ of the odometric position estimate. To do so, we assume that at the

starting point the initial covariance matrix Σ_p is known. For the motion increment $(\Delta s_r;\Delta s_l)$ we assume the following covariance matrix Σ_Δ:

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.8}$$

where Δs_r and Δs_l are the distances travelled by each wheel, and k_r, k_l are error constants representing the non-deterministic parameters of the motor drive and the wheel-floor interaction. As you can see in equation (5.8) we made the following assumption:

- The two errors of the individually driven wheels are independent[1]
- The errors are proportional to the absolute value of the traveled distances $(\Delta s_r;\Delta s_l)$.

These assumptions, while not perfect, are suitable and will thus be used for the further development of the error model. The *motion errors* are due to unprecise movement because of deformation of wheel, slippage, unequal floor, errors in encoders, *et cetera*. The values for the error constants k_r and k_l depend on the robot and the environment and should be experimentally established by performing and analyzing representative movements.

If we assume that p and $\Delta_{rl} = (\Delta s_r;\Delta s_l)$ are uncorrelated and the derivation of f (equ. (5.7)) is reasonably approximated by the first order Taylor expansion (linearization) we conclude, using the error propagation law (see section 4.2.3):

$$\Sigma_{p'} = \nabla_p f \cdot \Sigma_p \cdot \nabla_p f^T + \nabla_{\Delta_{rl}} f \cdot \Sigma_\Delta \cdot \nabla_{\Delta_{rl}} f^T \tag{5.9}$$

The covariance matrix Σ_p is, of course, always given by the $\Sigma_{p'}$ of the previous step, and can thus be calculated after specifying an initial value (e.g. 0).

Using equation (5.7) we can develop the two *Jacobians* $F_p = \nabla_p f$ and $F_{\Delta_{rl}} = \nabla_{\Delta_{rl}} f$:

$$F_p = \nabla_p f = \nabla_p (f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta / 2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta / 2) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

1. If there is more knowledge regarding the actual robot kinematics, the correlation terms of the covariance matrix could also be used.

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos(\theta + \Delta\theta/2) - \frac{\Delta s}{2b}\sin(\theta + \Delta\theta/2) & \frac{1}{2}\cos(\theta + \Delta\theta/2) + \frac{\Delta s}{2b}\sin(\theta + \Delta\theta/2) \\ \frac{1}{2}\sin(\theta + \Delta\theta/2) + \frac{\Delta s}{2b}\cos(\theta + \Delta\theta/2) & \frac{1}{2}\sin(\theta + \Delta\theta/2) - \frac{\Delta s}{2b}\cos(\theta + \Delta\theta/2) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \quad (5.11)$$

The details for arriving at equation (5.11) are:

$$F_{\Delta_{rl}} = \nabla_{\Delta_{rl}} f = \begin{bmatrix} \frac{\partial f}{\partial \Delta s_r} & \frac{\partial f}{\partial \Delta s_l} \end{bmatrix} = \ldots \quad (5.12)$$

$$\ldots = \begin{bmatrix} \frac{\partial \Delta s}{\partial \Delta s_r}\cos(\theta + \Delta\theta/2) + (-\sin(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta s}{\partial \Delta s_l}\cos(\theta + \Delta\theta/2) + (-\sin(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_l} \\ \frac{\partial \Delta s}{\partial \Delta s_r}\sin(\theta + \Delta\theta/2) + (\cos(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta s}{\partial \Delta s_l}\sin(\theta + \Delta\theta/2) + (\cos(\theta + \Delta\theta/2))\frac{\partial \Delta\theta}{\partial \Delta s_l} \\ \frac{\partial \Delta\theta}{\partial \Delta s_r} & \frac{\partial \Delta\theta}{\partial \Delta s_l} \end{bmatrix} \quad (5.13)$$

and with

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \quad ; \quad \Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \quad (5.14)$$

$$\frac{\partial \Delta s}{\partial \Delta s_r} = \frac{1}{2} \quad ; \quad \frac{\partial \Delta s}{\partial \Delta s_l} = \frac{1}{2} \quad ; \quad \frac{\partial \Delta\theta}{\partial \Delta s_r} = \frac{1}{b} \quad ; \quad \frac{\partial \Delta\theta}{\partial \Delta s_l} = -\frac{1}{b} \quad (5.15)$$

we obtain equation (5.11).

Figures 5.4 and 5.5 show typical examples of how the position errors grow with time. The results have been computed using the error model presented above.

Once the error model has been established, the error parameters must be specified. One can compensate for deterministic errors properly calibrating the robot. However the error parameters specifying the non-deterministic errors can only be quantified by statistical (repetitive) measurements. A detailed discussion of odometric errors and a method for calibration and quantification of deterministic and non-deterministic errors can be found in [4].

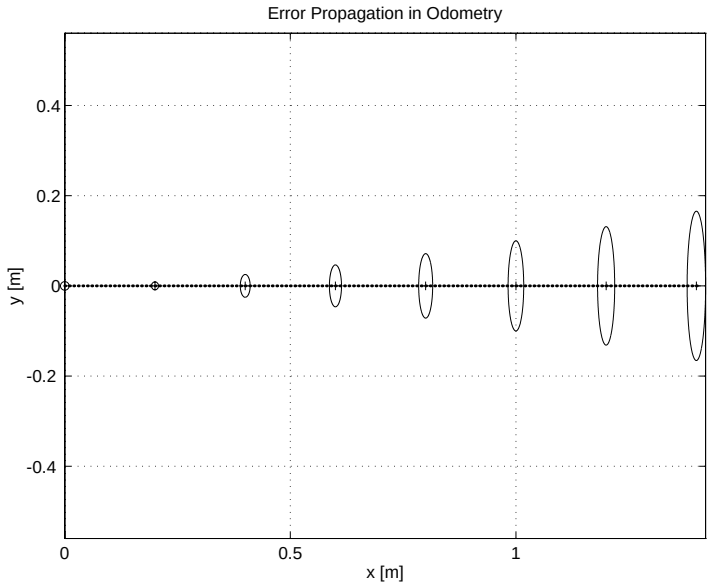


Fig 5.4 *Growth of the pose uncertainty for straight line movement: Note that the uncertainty in y grows much faster than in the direction of movement. This results from the integration of the uncertainty about the robot's orientation. The ellipses drawn around the robot positions represent the uncertainties in the x,y direction (e.g. 3σ). The uncertainty of the orientation θ is not represented in the picture although its effect can be indirectly observed.*

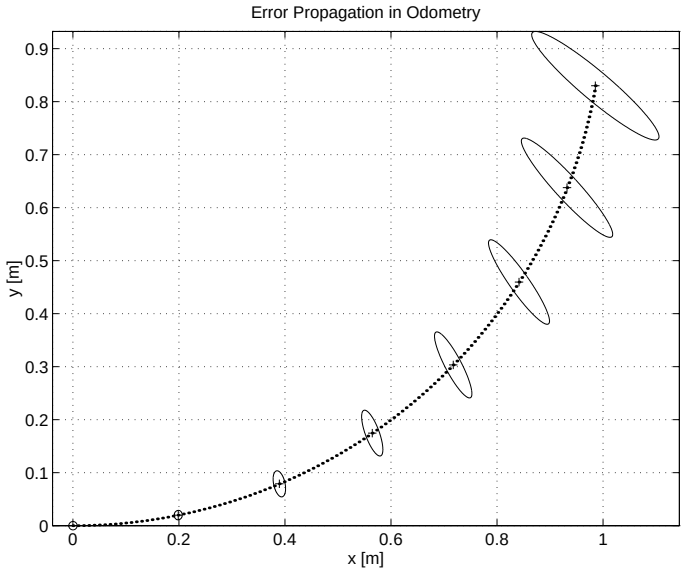


Fig 5.5 *Growth of the pose uncertainty for circular movement (r=const): Again, the uncertainty perpendicular to the movement grows much faster than that in the direction of movement. Note that the main axis of the uncertainty ellipsis does not remain perpendicular to the direction of movement.*

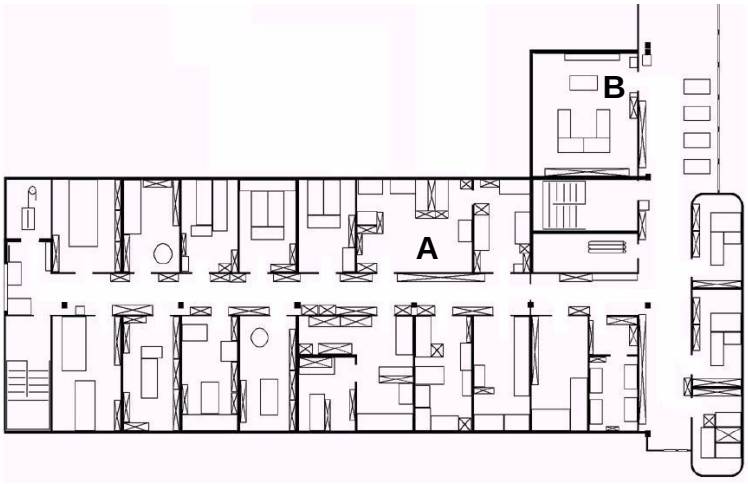


Fig 5.6 A Sample Environment

5.3 To Localize or Not to Localize: localization-based navigation versus programmed solutions

Figure 5.6 depicts a standard indoor environment that a mobile robot navigates. Suppose that the mobile robot in question must deliver messages between two specific rooms in this environment: rooms ***A*** and ***B***. In creating a navigation system, it is clear that the mobile robot will need sensors and a motion control system. Sensors are absolutely required to avoid hitting moving obstacles such as humans, and some motion control system is required so that the robot can deliberately move.

It is less evident, however, whether or not this mobile robot will require a *localization system*. Localization may seem mandatory in order to successfully navigate between the two rooms. It is through localizing on a map, after all, that the robot can hope to recover its position and detect when it has arrived at the goal location. It is true that, at the least, the robot must have a way of detecting the goal location. However, explicit localization with reference to a map is not the only strategy that qualifies as a goal detector.

An alternative, espoused by the behavior-based community, suggests that, since sensors and effectors are noisy and information-limited, one should avoid creating a geometric map for localization. Instead, this community suggests designing sets of behaviors that together result in the desired robot motion. Fundamentally, this approach avoids explicit reasoning about localization and position, and thus generally avoids explicit path planning as well.

This technique is based on a belief that there exists a procedural solution to the particular navigation problem at hand. For example, in Fig. 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 disadvan-

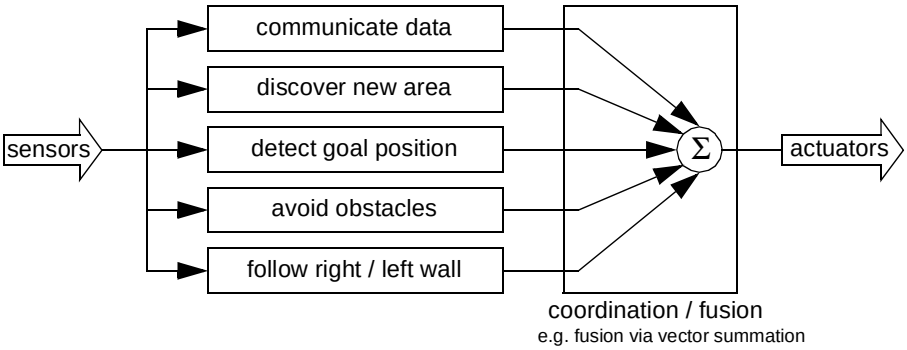


Fig 5.7 An Architecture for Behavior-based Navigation

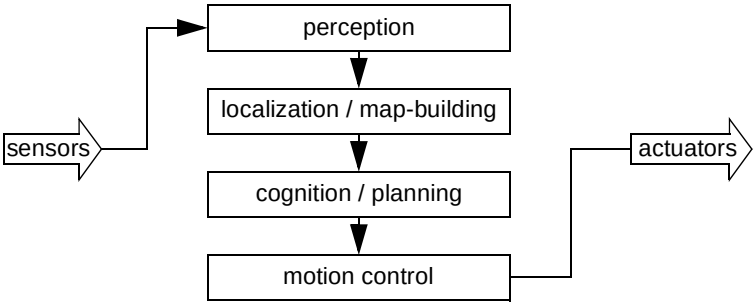


Fig 5.8 An Architecture for Map-based (or model-based) Navigation

tages, 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 re-tune all of the existing behaviors again to ensure that the new interactions with the freshly introduced behavior are all stable.

In contrast to the behavior-based approach, the map-based approach includes both *localization* and *cognition* modules (see Fig. 5.8). In map-based navigation, the robot explicitly attempts to localize by collecting sensor data, then updating some belief about its position with respect to a map of the environment. The key advantages of the map-based approach for navigation are as follows:

- The explicit, map-based concept of position makes the system's belief about position transparently available to the human operators.
- The existence of the map itself represents a medium for communication between human and robot: the human can simply give the robot a new map if the robot goes to

a new environment.

- 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 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, or 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 expressed as a single unique point on the map. In Fig. 5.10, three examples of a single hypothesis belief are shown using three different map representations of the same actual environment (fig. 5.10a). In 5.10b, a single point is geometrically annotated as the robot's position in a continuous two-dimensional geometric map. In 5.10c, the map is a discrete, tessellated map, and the position is noted at the same level of fidelity as the map cell size. In 5.10d, the map is not geometrical at all but abstract and topological. In this case, the single hypothesis of position involves identifying a single node *i* in the topological graph as the robot's position.

The principal advantage of the single hypothesis representation of position stems from the fact that, given a unique belief, there is no position ambiguity. The unambiguous nature of this representation facilitates decision-making at the robot's cognitive level (e.g. path planning). The robot can simply assume that its belief is correct, and can then select its future actions based on its unique position.

Just as decision-making is facilitated by a single-position hypothesis, so updating the robot's belief regarding position is also facilitated, since the single position must be updated by definition to a new, single position. The challenge with this position update approach, which ultimately is the principal disadvantage of single-hypothesis representation, is that robot motion often induces uncertainty due to effectory and sensory noise. Therefore, forcing the position update process to always generate a *single* hypothesis of position is challenging

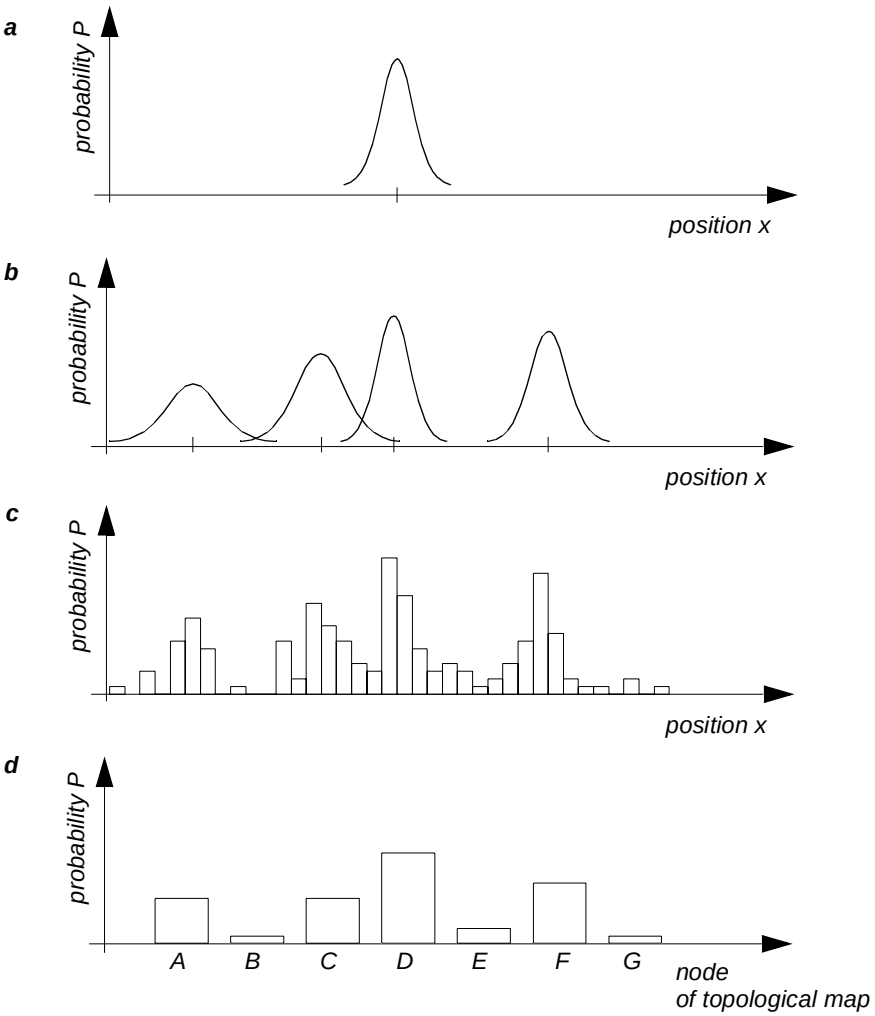


Fig 5.9 *Belief representation regarding the robot position (1 dimensional) in continuous and discretized (tessellated) maps.*
a)Continuous map with multiple hypothesis belief, e.g. single Gaussian centered at a single continuous value
b)Continuous map with multiple hypothesis belief, e.g. multiple Gaussians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

and, often, impossible.

5.4.2 Multiple Hypothesis Belief

In the case of multiple hypothesis beliefs regarding position, the robot tracks not just a single possible position but a possibly infinite set of positions.

In one simple example originating in the work of Jean-Claude Latombe [5, 89], the robot's position is described in terms of a convex polygon positioned in a two-dimensional map of the environment. This multiple hypothesis representation communicates the set of possible robot positions geometrically, with no preference ordering over the positions. Each point in

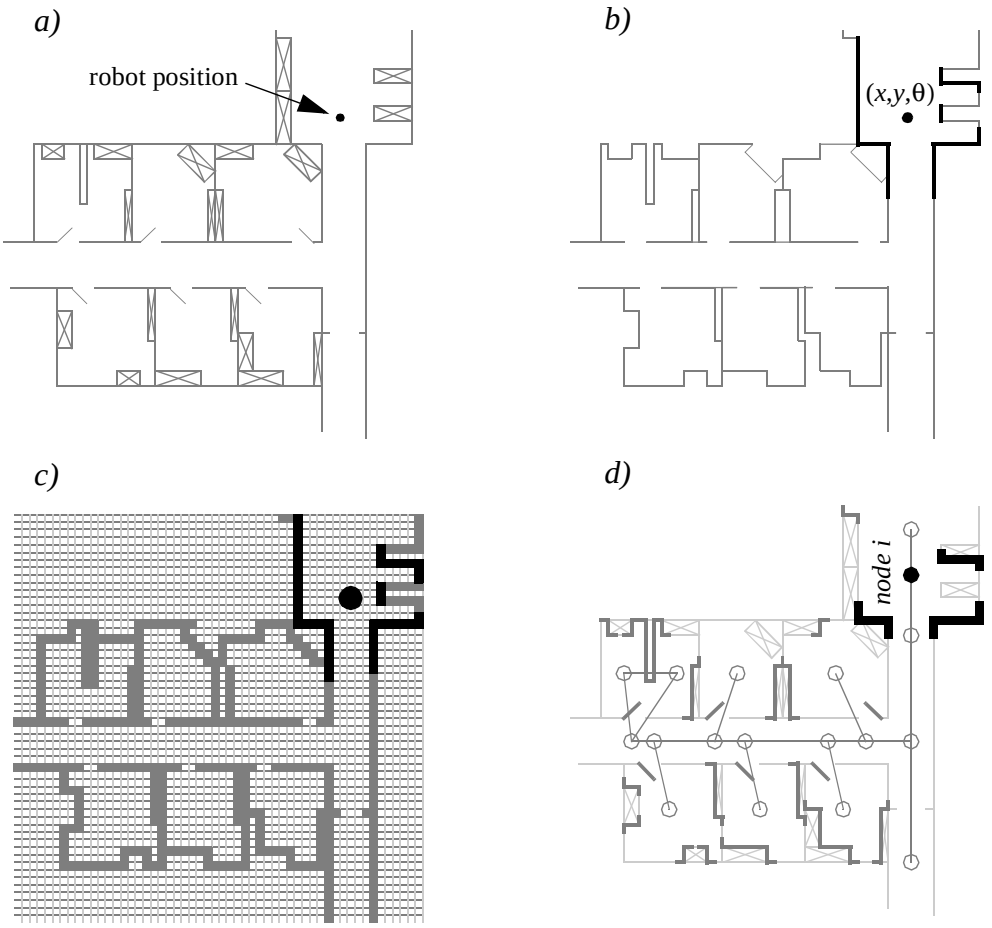
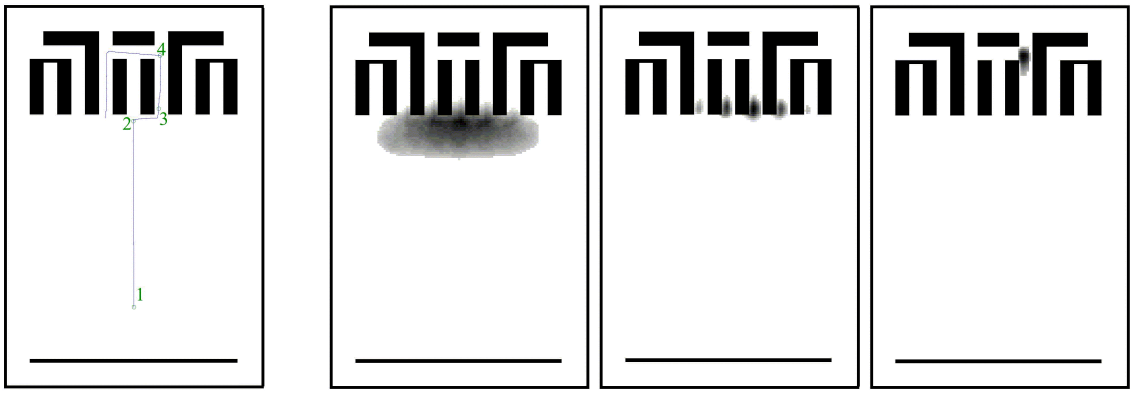


Fig 5.10 *Three examples of single hypotheses of position using different map representation.*
a) real map with walls, doors and furniture
b) line-based map
-> around 100 lines with two parameters
c) occupancy grid based map
-> around 3000 gird cells sizing 50x50 cm
d) topological map using line features (Z/S-lines) and doors
-> around 50 features and 18 nodes

the map is simply either contained by the polygon and, therefore, in the robot's belief set, or outside the polygon and thereby excluded. Mathematically, the position polygon serves to partition the space of possible robot positions. Such a polygonal representation of the multiple hypothesis belief can apply to a continuous, geometric map of the environment or, alternatively, to a tessellated, discrete approximation to the continuous environment.

It may be useful, however, to incorporate some ordering on the possible robot positions, capturing the fact that some robot positions are likelier than others. A strategy for representing a continuous multiple hypothesis belief state along with a preference ordering over possible positions is to model the belief as a mathematical distribution. For example, [42,47] notate the robot's position belief using an $\{X,Y\}$ point in the two-dimensional environment as the mean μ plus a standard deviation parameter σ, thereby defining a Gaussian distribution. The intended interpretation is that the distribution at each position represents the probability



Path of the robot *Belief states at positions 2, 3 and 4*

Fig 5.11 *Example of multiple hypothesis tracking (courtesy of W. Burgard [43]). The belief state that is largely distributed becomes very certain after moving to position 4*

assigned to the robot being at that location. This representation is particularly amenable to mathematically defined tracking functions, such as the Kalman Filter, that are designed to operate efficiently on Gaussian distributions.

An alternative is to represent the set of possible robot positions, not using a single Gaussian probability density function, but using discrete markers for each possible position. In this case, each possible robot position is individually noted along with a confidence or probability parameter (See Fig. (5.11)). In the case of a highly tessellated map this can result in thousands or even tens of thousands of possible robot positions in a single belief state.

The key advantage of the multiple hypothesis representation is that the robot can explicitly maintain uncertainty regarding its position. If the robot only acquires partial information regarding position from its sensors and effectors, that information can conceptually be incorporated in an updated belief.

A more subtle advantage of this approach revolves around the robot's ability to explicitly measure its own degree of uncertainty regarding position. This advantage is the key to a class of localization and navigation solutions in which the robot not only reasons about reaching a particular goal, but reasons about the future trajectory of its own belief state. For instance, a robot may choose paths that minimize its future position uncertainty. An example of this approach is [90], in which the robot plans a path from point *A* to *B* that takes it near a series of landmarks in order to mitigate localization difficulties. This type of explicit reasoning about the effect that trajectories will have on the quality of localization requires a multiple hypothesis representation.

One of the fundamental disadvantages of the multiple hypothesis approaches involves decision-making. If the robot represents its position as a region or set of possible positions, then how shall it decide what to do next? Figure 5.11 provides an example. At position 3, the robot's belief state is distributed among 5 hallways separately. If the goal of the robot is to travel down one particular hallway, then given this belief state what action should the robot choose?

The challenge occurs because some of the robot's possible positions imply a motion trajec-

tory that is inconsistent with some of its other possible positions. One approach that we will see in the case studies below is to assume, for decision-making purposes, that the robot is physically at the most probable location in its belief state, then to choose a path based on that current position. But this approach demands that each possible position have an associated probability.

In general, the right approach to such a decision-making problems would be to decide on trajectories that eliminate the ambiguity explicitly. But this leads us to the second major disadvantage of the multiple hypothesis approaches. In the most general case, they can be computationally very expensive. When one reasons in a three dimensional space of discrete possible positions, the number of possible belief states in the single hypothesis case is limited to the number of possible positions in the 3D world. Consider this number to be N. When one moves to an arbitrary multiple hypothesis representation, then the number of possible belief states is the power set of N, which is far larger: 2^N. Thus explicit reasoning about the possible trajectory of the belief state over time quickly becomes computationally untenable as the size of the environment grows.

There are, however, specific forms of multiple hypothesis representations that are somewhat more constrained, thereby avoiding the computational explosion while allowing a limited type of multiple hypothesis belief. For example, if one assumes a Gaussian distribution of probability centered at a single position, then the problem of representation and tracking of belief becomes equivalent to Kalman Filtering, a straightforward mathematical process described below. Alternatively, a highly tessellated map representation combined with a limit of 10 possible positions in the belief state, results in a discrete update cycle that is, at worst, only 10x more computationally expensive than single hypothesis belief update.

In conclusion, the most critical benefit of the multiple hypothesis belief state is the ability to maintain a sense of position while explicitly annotating the robot's uncertainty about its own position. This powerful representation has enabled robots with limited sensory information to navigate robustly in an array of environments, as we shall see in the case studies below.

5.5 Map Representation

The problem of representing the environment in which the robot moves is a dual of the problem of representing the robot's possible position or positions. Decisions made regarding the environmental representation can have impact on the choices available for robot position representation. Often the fidelity of the position representation is bounded by the fidelity of the map.

Three fundamental relationships must be understood when choosing a particular map representation:

- The precision of the map must appropriately match the precision with which the robot needs to achieve its goals.
- The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors.
- The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization and navigation.

In the following sections, we identify and discuss critical design choices in creating a map representation. Each such choice has great impact on the relationships listed above and on the resulting robot localization architecture. As we will see, the choice of possible map representations is broad. Selecting an appropriate representation requires understanding all of the trade-offs inherent in that choice as well as understanding the specific context in which a particular mobile robot implementation must perform localization. In general, the environment representation and model can be roughly classified as presented in chapter 4.3.

5.5.1 Continuous Representations

A continuous-valued map is one method for *exact* decomposition of the environment. The position of environmental features can be annoted precisely in continuous space. Mobile robot implementations to date use continuous maps only in two dimensional representations, as further dimensionality can result in computational explosion.

A common approach is to combine the exactness of a continuous representation with the compactness of the *closed world assumption*. This means that one assumes that the representation will specify all environmental objects in the map, and that any area in the map that is devoid of objects has no objects in the corresponding portion of the environment. Thus, the total storage needed in the map is proportional to the density of objects in the environment, and a sparse environment can be represented by a low-memory map.

One example of such a representation, shown in Figure 5.12, is a 2D representation in which polygons represent all obstacles in a continuos-valued coordinate space. This is similar to the method used by Latombe [5, 113] and others to represent environments for mobile robot path planning techniques.

In the case of [5, 113], most of the experiments are in fact simulations run exclusively within the computer's memory. Therefore, no real effort would have been expended to attempt to use sets of polygons to describe a real-world environment, such as a park or office building.

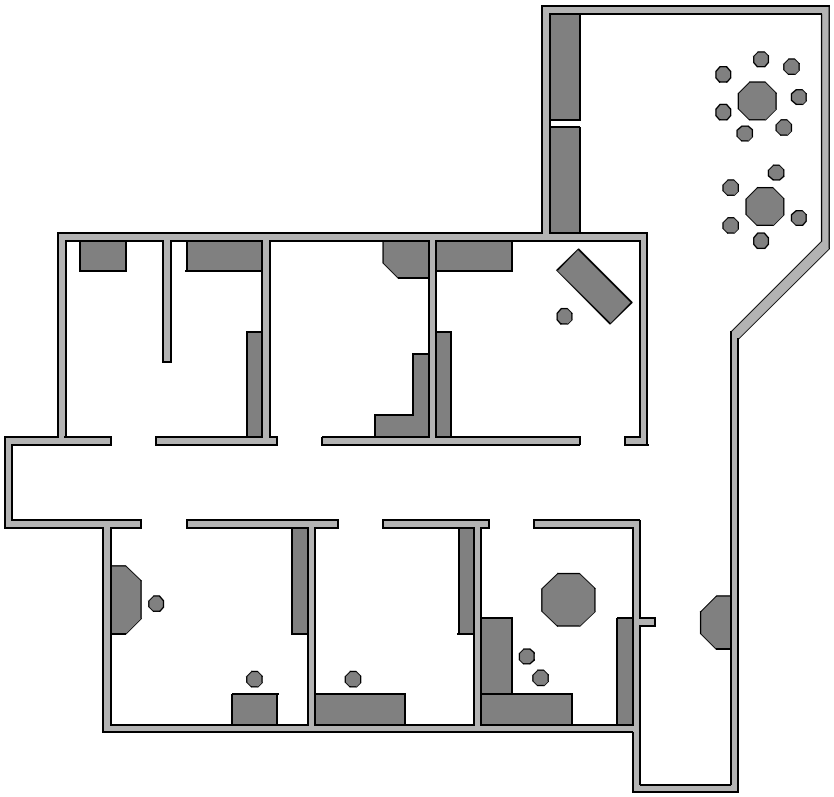


Fig 5.12 A continous representation using polygons as environmental obstacles

In other work in which real environments must be captured by the maps, one sees a trend toward selectivity and abstraction. The human map-maker tends to capture on the map, for localization purposes, only objects that can be detected by the robot's sensors and, furthermore, only a subset of the features of real-world objects.

It should be immediately apparent that geometric maps can capably represent the physical locations of objects without referring to their texture, color, elasticity, or any other such secondary features that do not related directly to position and space. In addition to this level of simplification, a mobile robot map can further reduce memory usage by capturing only aspects of object geometry that are immediately relevant to localization. For example all objects may be approximated using very simple convex polygons, sacrificing map felicity for the sake of computational speed.

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 strikes. 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 repre-

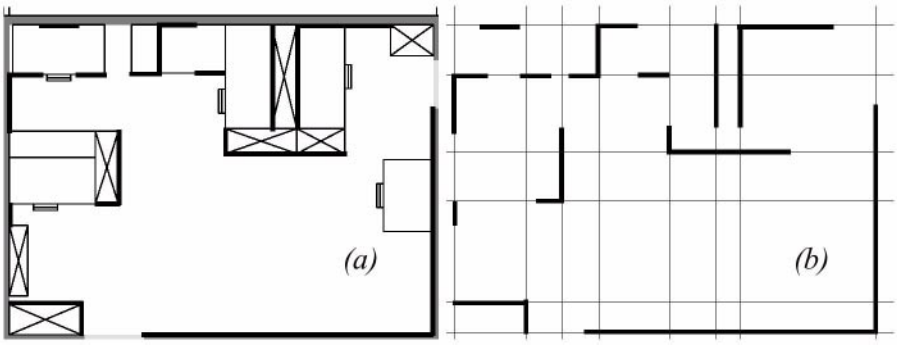


Fig 5.13 *Example of a continuous-valued line representation of EPFL.*
left: real map
right: representation with a set of infinite lines

sentation. 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 continuos 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 5.30, in which the position of the robot is depicted by an oval bounding area.

Yet, the continuous representation does not disallow representation of position in the form of a discrete set of possible positions. For instance, in [111] the robot position belief state is captured by sampling nine continuous-valued positions from within a region near the robot's best known position. This algorithm captures, within a continuous space, a discrete sampling of possible robot positions.

In summary, the key advantage of a continuous map representation is the potential for high accuracy and expressiveness with respect to the environmental configuration as well as the robot position within that environment. The danger of a continuous representation is that the map may be computationally costly. But this danger can be tempered by employing abstraction and capturing only the most relevant environmental features. Together with the use of the *closed world assumption*, these techniques can enable a continuous-valued map to be no more costly, and sometimes even less costly, than a standard discrete representation.

5.5.2 Decomposition Strategies

In the section above, we discussed one method of simplification, in which the continuous map representation contains a set of infinite lines that approximate real-world environmental lines based on a two-dimensional slice of the world. Basically this transformation from the real world to the map representation is a filter that removes all non-straight data and furthermore extends line segment data into infinite lines that require fewer parameters.

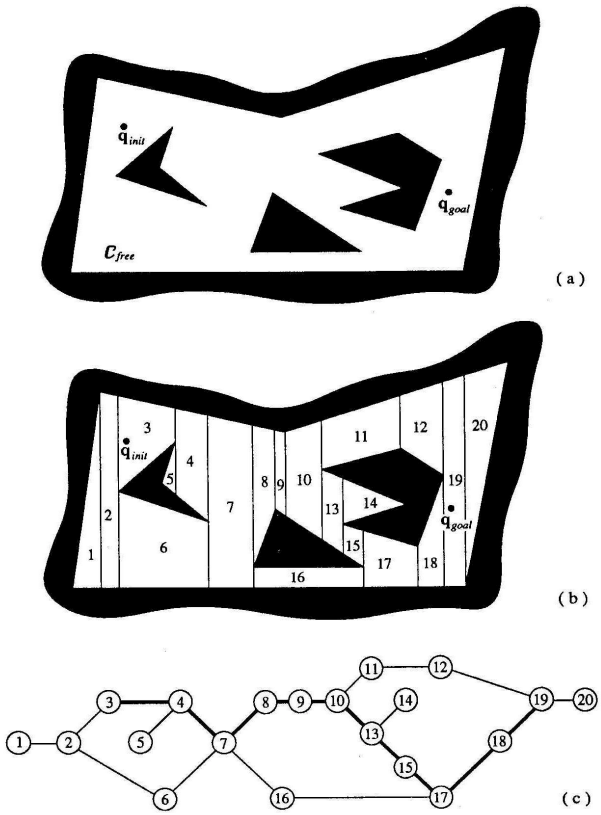


Fig 5.14 *Example of exact cell decomposition.*

A more dramatic form of simplification is *abstraction*: a general decomposition and selection of environmental features. In this section, we explore decomposition as applied in its more extreme forms to the question of map representation.

Why would one radically decompose the real environment during the design of a map representation? The immediate disadvantage of decomposition and abstraction is the loss of fidelity between the map and the real world. Both qualitatively, in terms of overall structure, and quantitatively, in terms of geometric precision, a highly abstract map does not compare favorably to a high-fidelity map.

Despite this disadvantage, decomposition and abstraction may be useful if the abstraction can be planned carefully so as to capture the relevant, *useful* features of the world while discarding all other features. The advantage of this approach is that the map representation can potentially be minimized. Furthermore, if the decomposition is hierarchical, such as in a pyramid of recursive abstraction, then reasoning and planning with respect to the map representation may be computationally far superior to planning in a fully detailed world model.

A standard, lossless form of *opportunistic decomposition* is termed *exact cell decomposition.* This method, introduced by [5], achieves decomposition by selecting boundaries between discrete cells based on geometric criticality.

Figure 5.14 depicts an exact decomposition of a planar workspace populated by polygonal obstacles. The map representation tessellates the space into areas of free space. The representation can be extremely compact because each such area is actually stored as a single node, shown in the graph at the bottom of Figure 5.14.

The underlying assumption behind this decomposition is that the particular position of a ro-

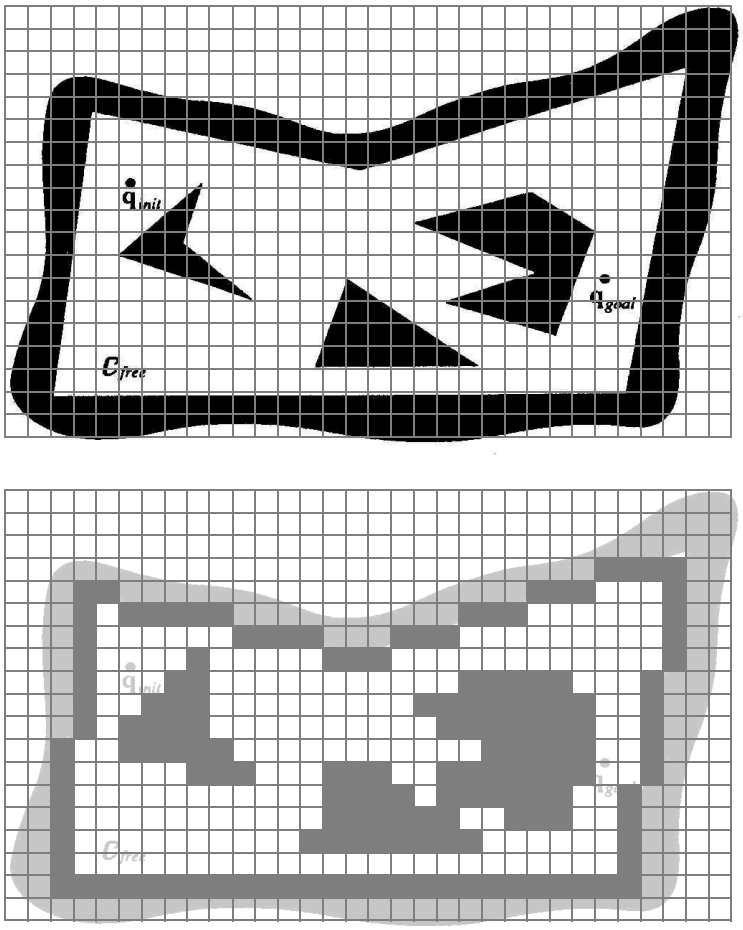


Fig 5.15 Fixed decomposition of the same space. (narrow passage disappears)

bot within each area of free space does not matter. What matters is the robot's ability to traverse from each area of free space to the adjacent areas. Therefore, as with other representations we will see, the resulting graph captures the adjacency of map locales. If indeed the assumptions are valid and the robot does not care about its precise position within a single area, then this can be an effective representation that nonetheless captures the connectivity of the environment.

Such an exact decomposition is not always appropriate. Exact decomposition is a function of the particular environment obstacles and free space. If this information is expensive to collect or even unknown, then such an approach is not feasible.

An alternative is *fixed decomposition*, in which the world is tessellated, transforming the continuos real environment into a discrete approximation for the map. Such a transformation is demonstrated in Figure 5.15, which depicts what happens to obstacle-filled and free areas during this transformation. The key disadvantage of this approach stems from its *inexact* nature. It is possible for narrow passageways to be lost during such a transformation, as shown in Figure 5.15. Formally this means that fixed decomposition is sound but not complete. Yet another approach is adaptive cell decomposition as presented in Figure 5.16.

The concept of fixed decomposition is extremely popular in mobile robotics; it is perhaps the single most common map representation technique currently utilized. One very popular

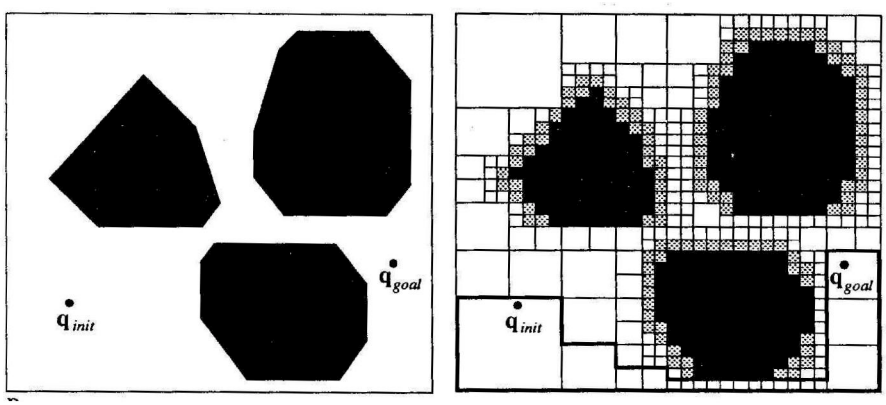


Fig 5.16 *Example of adaptive decomposition of an environment.*

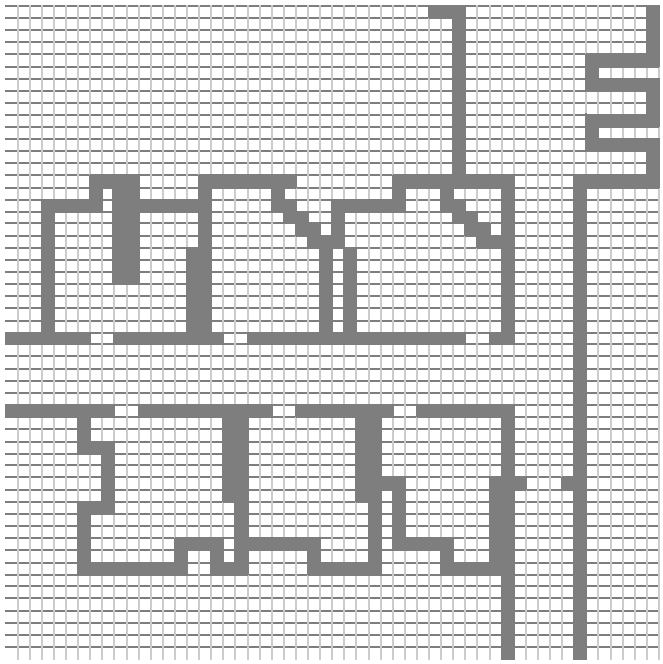


Fig 5.17 *Example of an occupancy grid map representation.*

version of fixed decomposition is known as the *occupancy grid* representation [91]. In an occupancy grid, the environment is represented by a discrete grid, where each cell is either filled (part of an obstacle) or empty (part of free space). This method is of particular value when a robot is equipped with range-based sensors because the range values of each sensor, combined with the absolute position of the robot, can be used directly to update the filled/ empty value of each cell.

In the occupancy grid, each cell may have a counter, whereby the value 0 indicates that the cell has not been "hit" by any ranging measurements and, therefore, it is likely free space. As the number of ranging strikes increases, the cell's value is incremented and, above a certain threshold, the cell is deemed to be an obstacle. By discounting the values of cells over time, both hysteresis and the possibility of transient obstacles can be represented using this occupancy grid approach. Figure 5.17 depicts an occupancy grid representation in which the darkness of each cell is proportional to the value of its counter. One commercial robot that uses a standard occupancy grid for mapping and navigation is the Cye robot [112].

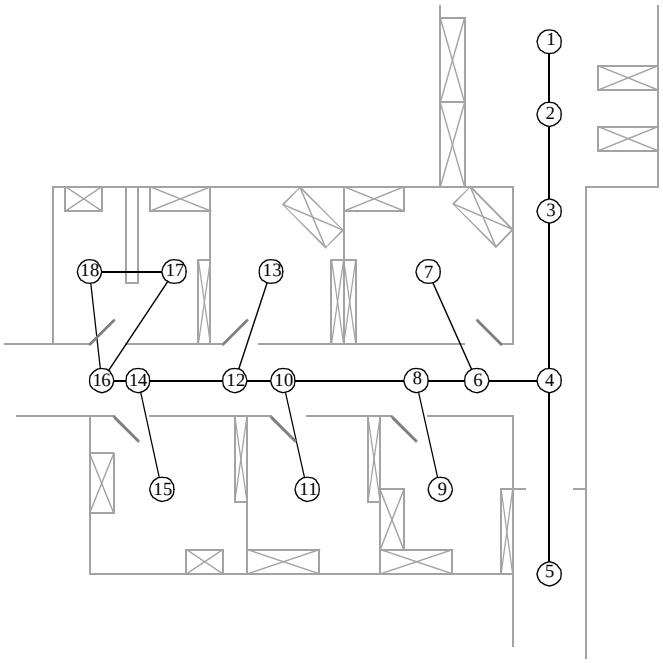


Fig 5.18 *A topological representation of an indoor office area.*

There remain two main disadvantages of the occupancy grid approach. First, the size of the map in robot memory grows with the size of the environment and, if a small cell size is used, this size can quickly become untenable. This occupancy grid approach is not compatible with the *closed world assumption*, which enabled continuous representations to have potentially very small memory requirements in large, sparse environments. In contrast, the occupancy grid must have memory set aside for every cell in the matrix. Furthermore, any fixed decomposition method such as this imposes a geometric grid on the world *a priori*, regardless of the environmental details. This can be inappropriate in cases where geometry is not the most salient feature of the environment.

For these reasons, an alternative, called *topological* decomposition, has been the subject of some exploration in mobile robotics. Topological approaches avoid direct measurement of geometric environmental qualities, instead concentrating on characteristics of the environment that are most relevant to the robot for localization.

Formally, a topological representation is a graph that specifies two things: *nodes* and the *connectivity* between those nodes. Insofar as a topological representation is intended for the use of a mobile robot, nodes are used to denote areas in the world and arcs are used to denote adjacency of pairs of nodes. When an arc connects two nodes, then the robot can traverse from one node to the other without requiring traversal of any other intermediary node.

Adjacency is clearly at the heart of the topological approach, just as adjacency in a cell decomposition representation maps to geometric adjacency in the real world. However, the topological approach diverges in that the nodes are not of fixed size nor even specifications of free space. Instead, nodes document an area based on any sensor discriminant such that the robot can recognize entry and exit of the node.

Figure 5.18 depicts a topological representation of a set of hallways and offices in an indoor

environment. In this case, the robot is assumed to have an intersection detector, perhaps using sonar and vision to find intersections between halls and between halls and rooms. Note that nodes capture geometric space and arcs in this representation simply represent connectivity.

Another example of topological representation is the work of Dudek [49], 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 Dudek's 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 the potential to be both compact like a topological representation and precise as with all direct geometric representations.

Yet, a chief motivation of the topological approach is that the environment may contain important non-geometric features - features that have no ranging relevance but are useful for localization. In Chapter 4 we described such whole-image vision-based features.

In contrast to these whole-image feature extractors, often spatially localized landmarks are artificially placed in an environment to impose a particular visual-topological connectivity upon the environment. In effect, the artificial landmark can impose artificial structure. Examples of working systems operating with this landmark-based strategy have also demonstrated success. Latombe's landmark-based navigation research [89] has been implemented on real-world indoor mobile robots that employ paper landmarks attached to the ceiling as the locally observable features. Chips the museum robot is another robot that uses man-made landmarks to obviate the localization problem. In this case, a bright pink square serves as a landmark with dimensions and color signature that would be hard to accidentally reproduce in a museum environment [88]. One such museum landmark is shown in Figure (5.19).

In summary, range is clearly not the only measurable and useful environmental value for a mobile robot. This is particularly true due to the advent of color vision as well as laser rangefinding, which provides reflectance information in addition to range information. Choosing a map representation for a particular mobile robot requires first understanding the sensors available on the mobile robot and second understanding the mobile robot's functional requirements (*e.g.* required goal precision and accuracy).

5.5.3 State of the Art: Current Challenges in Map Representation

The sections above describe major design decisions in regards to map representation choic-

Fig 5.19 An artificial landmark used by Chips during autonomous docking.

es. There are, however, fundamental real-world features that mobile robot map representations do not yet represent well. These continue to be the subject of open research, and several such challenges are described below.

The real world is dynamic. As mobile robots come to inhabit the same spaces as humans, they will encounter moving people, cars, strollers and the transient obstacles placed and moved by humans as they go about their activities. This is particularly true when one considers the home environment with which domestic robots will someday need to contend.

The map representations described above do not, in general, have explicit facilities for identifying and distinguishing between permanent obstacles (e.g. walls, doorways, etc.) and transient obstacles (e.g. humans, shipping packages, etc.). The current state of the art in terms of mobile robot sensors is partly to blame for this shortcoming. Although vision research is rapidly advancing, robust sensors that discriminate between moving animals and static structures *from a moving reference frame* are not yet available. Furthermore, estimating the motion vector of transient objects remains a research problem.

Usually, the assumption behind the above map representations is that all objects on the map are effectively static. Partial success can be achieved by discounting mapped objects over time. For example, occupancy grid techniques can be more robust to dynamic settings by introducing temporal discounting, effectively treating transient obstacles as noise. The more challenging process of map creation is particularly fragile to environment dynamics; most mapping techniques generally require that the environment be free of moving objects during the mapping process. One exception to this limitation involves topological representations. Because precise geometry is not important, transient objects have little effect on the mapping or localization process, subject to the critical constraint that the transient objects must not change the topological connectivity of the environment. Still, neither the occupancy grid representation nor a topological approach is actively recognizing and representing transient

objects as distinct from both sensor error and permanent map features.

As vision sensing provides more robust and more informative content regarding the transience and motion details of objects in the world, mobile roboticists will in time propose representations that make use of that information. A classic example involves occlusion by human crowds. Museum tour guide robots generally suffer from an extreme amount of occlusion. If the robot's sensing suite is located along the robot's body, then the robot is effectively blind when a group of human visitors completely surrounds the robot. This is because its map contains only environment features that are, at that point, fully hidden from the robot's sensors by the wall of people. In the best case, the robot should recognize its occlusion and make no effort to localize using these invalid sensor readings. In the worst case, the robot will localize with the fully occluded data, and will update its location incorrectly. A vision sensor that can discriminate the local conditions of the robot (*e.g. we are surrounded by people*) can help eliminate this error mode.

A second open challenge in mobile robot localization involves the traversal of open spaces. Existing localization techniques generally depend on local measures such as range, thereby demanding environments that are somewhat densely filled with objects that the sensors can detect and measure. Wide open spaces such as parking lots, fields of grass and indoor atriums such as those found in convention centers pose a difficulty for such systems due to their relative sparseness. Indeed, when populated with humans, the challenge is exacerbated because any mapped objects are almost certain to be occluded from view by the people.

Once again, more recent technologies provide some hope for overcoming these limitations. Both vision and state-of-the-art laser rangefinding devices offer outdoor performance with ranges of up to a hundred meters and more. Of course, GPS performs even better. Such long-range sensing may be required for robots to localize using distant features.

This trend teases out a hidden assumption underlying most topological map representations. Usually, topological representations make assumptions regarding spatial locality: a node contains objects and features that are themselves within that node. The process of map creation thus involves making nodes that are, in their own self-contained way, recognizable by virtue of the objects contained within the node. Therefore, in an indoor environment, each room can be a separate node, and this is reasonable because each room will have a layout and a set of belongings that are unique to that room.

However, consider the outdoor world of a wide-open park. Where should a single node end and the next node begin? The answer is unclear because objects that are far away from the current node, or position, can yield information for the localization process. For example, the hump of a hill at the horizon, the position of a river in the valley and the trajectory of the sun all are non-local features that have great bearing on one's ability to infer current position. The spatial locality assumption is violated and, instead, replaced by a visibility criterion: the node or cell may need a mechanism for representing objects that are measurable and visible from that cell. Once again, as sensors improve and, in this case, as outdoor locomotion mechanisms improve, there will be greater urgency to solve problems associated with localization in wide-open settings, with and without GPS-type global localization sensors.

We end this section with one final open challenge that represents one of the fundamental academic research questions of robotics: sensor fusion. A variety of measurement types are possible using off-the-shelf robot sensors, including heat, range, acoustic and light-based reflectivity, color, texture, friction, etc. Sensor fusion is a research topic closely related to map representation. Just as a map must embody an environment in sufficient detail for a robot to perform localization and reasoning, sensor fusion demands a representation of the world that is sufficiently general and expressive that a variety of sensor types can have their data correlated appropriately, strengthening the resulting percepts well beyond that of any individual sensor's readings.

Perhaps the only general implementation of sensor fusion to date is that of neural network classifier. Using this technique, any number and any type of sensor values may be jointly combined in a network that will use whatever means necessary to optimize its classification accuracy. For the mobile robot that must use a human-readable internal map representation, no equally general sensor fusion scheme has yet been born. It is reasonable to expect that, when the sensor fusion problem is solved, integration of a large number of disparate sensor types may easily result in sufficient discriminatory power for robots to achieve real-world navigation, even in wide-open and dynamic circumstances such as a public square filled with people.

5.6 Probabilistic Map-Based Localization

5.6.1 Introduction

As stated earlier, multiple hypothesis position representation is advantageous because the robot can explicitly track its own beliefs regarding its possible positions in the environment. Ideally, the robot's *belief state* will change, over time, as is consistent with its motor outputs and perceptual inputs. One geometric approach to multiple hypothesis representation, mentioned earlier, involves identifying the possible positions of the robot by specifying a polygon in the environmental representation [113]. This method does not provide any indication of the relative chances between various possible robot positions.

Probabilistic techniques differ from this because they explicitly identify probabilities with the possible robot positions, and for this reason these methods have been the focus of recent research. In the following sections we present two classes of probabilistic localization. The first class, *Markov localization*, uses an explicitly specified probability distribution across all possible robots positions. The second method, *Kalman filter localization*, uses a Gaussian probability density representation of robot position and scan matching for localization. Unlike Markov localization, Kalman filter localization does not independently consider each possible pose in the robot's configuration space. Interestingly, the Kalman filter localization process results from the Markov localization axioms if the robot's position uncertainty is assumed to have a Gaussian form [28 page 43-44].

Before discussing each method in detail, we present the general robot localization problem and solution strategy. Consider a mobile robot moving in a known environment. As it starts to move, say from a precisely known location, it can keep track of its motion using odometry. Due to odometry uncertainty, after some movement the robot will become very uncertain about its position (see section 5.2.4). To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot might use its on-board sensors (ultrasonic, range sensor, vision) to make observations of its environment. The information provided by the robot's odometry, plus the information provided by such exteroceptive observations can be combined to enable the robot to localize as well as possible with respect to its map. The processes of updating based on proprioceptive sensor values and exteroceptive sensor values are often separated logically, leading to a general two-step process for robot position update.

Action update represents the application of some action model *Act* to the mobile robot's proprioceptive encoder measurements o_t and prior belief state s_{t-1} to yield a new belief state representing the robot's belief about its current position. Note that throughout this chapter we will assume that the robot's proprioceptive encoder measurements are used as the best possible measure of its actions over time. If, for instance, a differential drive robot had motors without encoders connected to its wheels and employed open-loop control, then instead of encoder measurements the robot's highly uncertain estimates of wheel spin would need to be incorporated. We ignore such cases and therefore have a simple formula:

$$s'_t = Act(o_t, s_{t-1}). \tag{5.16}$$

Perception update represents the application of some perception model *See* to the mobile robot's exteroceptive sensor inputs i_t and updated belief state s'_t to yield a refined belief state representing the robot's current position:

$$s_t = See(i_t, s'_t) \tag{5.17}$$

The perception model *See* and sometimes the action model *Act* are abstract functions of both the map and the robot's physical configuration (e.g. sensors and their positions, kinematics, etc.).

In general, the action update process contributes uncertainty to the robot's belief about position: encoders have error and therefore motion is somewhat nondeterministic. By contrast, perception update generally refines the belief state. Sensor measurements, when compared to the robot's environmental model, tend to provide clues regarding the robot's possible position.

In the case of Markov localization, the robot's belief state is usually represented as separate probability assignments for every possible robot pose in its map. The action update and perception update processes must update the probability of every cell in this case. Kalman filter localization represents the robot's belief state using a singe, well-defined Gaussian probability density function, and thus retains just a μ and σ parameterization of the robot's belief about position with respect to the map. Updating the parameters of the Gaussian distribution is all that is required. This fundamental difference in the representation of belief state leads to the following advantages and disadvantages of the two methods, as presented in [44]:

- Markov localization allows for localization starting from any unknown position and can thus recover from ambiguous situations because the robot can track multiple, completely disparate possible positions. However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and computational power can thus limit precision and map size.
- Kalman filter localization tracks the robot from an initially known position and is inherently both precise and efficient. In particular, Kalman filter localization can be used in continuous world representations. However, if the uncertainty of the robot becomes too large (e.g. due to a robot collision with an object) and thus not truly unimodal, the Kalman filter can fail to capture the multitude of possible robot positions and can become irrevocably lost.

In recent research projects improvements are achieved or proposed by either only updating the state space of interest within the Markov approach [43] or by combining both methods to create a hybrid localization system [44]. In the next two subsections we will present each approach in detail.

5.6.2 Markov Localization (see also [42, 45, 71, 72])

Markov localization tracks the robot's belief state using an arbitrary probability density function to represent the robot's position. In practice, all known Markov localization sys-

tems implement this generic belief representation by first tessellating the robot configuration space into a finite, discrete number of possible robot poses in the map. In actual applications, the number of possible poses can range from several hundred positions to millions of positions.

Given such a generic conception of robot position, a powerful update mechanism is required that can compute the belief state that results when new information (e.g. encoder values and sensor values) is incorporated into a prior belief state with arbitrary probability density. The solution is born out of probability theory, and so the next section describes the foundations of probability theory that apply to this problem, notably Bayes formula. Then, two subsequent subsections provide case studies, one robot implementing a simple feature-driven topological representation of the environment [45, 71, 72] and the other using a geometric grid-based map[42, 43].

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 B 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 Bayes formula for computing $p(A|B)$:

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

We use 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 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 $p(r=l)$:

$$p(l|i) = \frac{p(i|l)p(l)}{p(i)} \tag{5.21}$$

The value of $p(i|l)$ is key to Equation 5.21, and this probability of a sensor input at each robot position must be computed using some model. An obvious strategy would be to consult the robot's map, identifying the probability of particular sensor readings with each possible map position, given knowledge about the robot's sensor geometry and the mapped environment. The value of $p(l)$ is easy to recover in this case. It is simply the probability $p(r=l)$ associated with the belief state before the perceptual update process. Finally, note that the denominator $p(i)$ does not depend upon l; that is, as we apply Equation 5.21 to all positions l in L, the denominator never varies. Because it is effectively constant, in practice this denominator is usually dropped and, at the end of the perception update step, all probabilities in the belief state are re-normalized to sum at 1.0.

Now consider the *Act* function of Equation 5.16. *Act* maps a former belief state and encoder measurement (i.e. robot action) to a new belief state. In order to compute the probability of position l in the new belief state, one must integrate over all the possible ways in which the robot may have reached l according to the potential positions expressed in the former belief state. This is subtle but fundamentally important. The same location l can be reached from multiple source locations with the same encoder measurement o because the encoder measurement is uncertain. Temporal indices are required in this update equation:

$$p(l_t|o_t) = \int p(l_t|l'_{t-1}, o_t)p(l'_{t-1})dl'_{t-1} \tag{5.22}$$

Thus, the total probability for a specific position l is built up from the individual contributions from every location l' in the former belief state given encoder measurement o.

Equations 5.21 and 5.22 form the basis of Markov localization, and they incorporate the *Markov assumption.* Formally, this means that their output is a function only of the robot's previous state and its most recent actions (odometry) and perception. In a general, non-Markovian situation, the state of a system depends upon all of its history. After all, the value of a robot's sensors at time t do not really depend only on its position at time t. They depend to some degree on the trajectory of the robot over time; indeed on the entire history of the robot. For example, the robot could have experienced a serious collision recently that has biased the sensor's behavior. By the same token, the position of the robot at time t does not

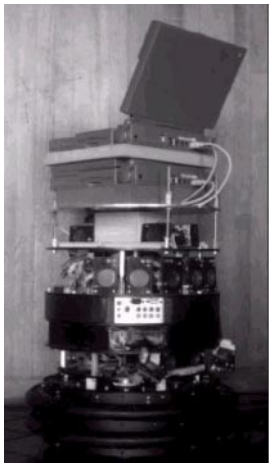


Fig 5.20 Dervish exploring its environment

really depend only on its position at time *t-1* and its odometric measurements. Due to its history of motion, one wheel may have worn more than the other, causing a left-turning bias over time that affects its current position.

So the Markov assumption is, of course, not a valid assumption. However the Markov assumption greatly simplifies tracking, reasoning and planning and so it is an approximation that continues to be extremely popular in mobile robotics.

5.6.2.2 Case Study I: Markov Localization using a Topological Map

A straightforward application of Markov localization is possible when the robot's environment representation already provides an appropriate decomposition. This is the case when the environment representation is purely topological.

Consider a contest in which each robot is to receive a topological description of the environment. The description would describe only the connectivity of hallways and rooms, with no mention of geometric distance. In addition, this supplied *map* would be imperfect, containing several false arcs (e.g. a closed door). Such was the case for the 1994 AAAI National Robot Contest, at which each robot's mission was to use the supplied map and its own sensors to navigate from a chosen starting position to a target room.

Dervish, the winner of this contest, employed probabilistic Markov localization and used just this multiple hypothesis belief state over a topological environmental representation. We now describe Dervish as an example of a robot with a topological representation and a probabilistic localization algorithm.

Dervish, shown in Figure 5.20, includes a sonar arrangement custom-designed for the 1994 AAAI National Robot Contest. The environment in this contest consisted of a rectilinear indoor office space filled with real office furniture as obstacles. Traditional sonars are arranged radially around the robot in a ring. Robots with such sensor configurations are subject to both tripping over short objects below the ring and to decapitation by tall objects (such as ledges, shelves and tables) that are above the ring.

Dervish's answer to this challenge was to arrange one pair of sonars diagonally upward to

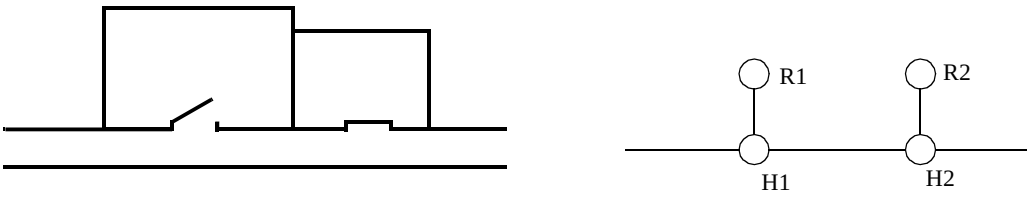


Fig 5.21 A geometric office environment (left) and its topological analogue (right)

detect ledges and other overhangs. In addition, the diagonal sonar pair also proved to ably detect tables, enabling the robot to avoid wandering underneath tall tables. The remaining sonars were clustered in sets of sonars, such that each individual transducer in the set would be at a slightly varied angle to minimize specularity. Finally, two sonars near the robot's base were able to detect low obstacles such as paper cups on the floor.

We have already noted that the representation provided by the contest organizers was purely topological, noting the connectivity of hallways and rooms in the office environment. Thus, it would be appropriate to design Dervish's perceptual system to detect matching perceptual events: the detection and passage of connections between hallways and offices.

This *abstract* perceptual system was implemented by viewing the trajectory of sonar strikes to the left and right sides of Dervish over time. Interestingly, this perceptual system would use time alone and no concept of encoder value in order to trigger perceptual events. Thus, for instance, when the robot detects a 7 to 17 cm indentation in the width of the hallway for more than one second continuously, a *closed door* sensory event is triggered. If the sonar strikes jump well beyond 17 cm for more than one second, an *open door* sensory event triggers.

Sonars have a notoriously problematic error mode known as specular reflection: when the sonar unit strikes a flat surface at a shallow angle, the sound may reflect coherently away from the transducer, resulting in a large overestimate of range. Dervish was able to filter such potential noise by tracking its approximate angle in the hallway and completely suppressing sensor events when its angle to the hallway parallel exceeded 9 degrees. Interestingly, this would result in a conservative perceptual system that would easily miss features because of this suppression mechanism, particularly when the hallway is crowded with obstacles that Dervish must negotiate. Once again, the conservative nature of the perceptual system, and in particular its tendency to issue false negatives, would point to a probabilistic solution to the localization problem so that a complete trajectory of perceptual inputs could be considered.

Dervish's environment representation was a classical topological map, identical in abstraction and information to the map provided by the contest organizers. Figure 5.21 depicts a geometric representation of a typical office environment and the topological map for the same office environment. One can place nodes at each intersection and in each room, resulting in the case of figure 5.21 with four nodes total.

Once again, though, it is crucial that one maximize the information content of the representation based on the available percepts. This means reformulating the standard topological graph shown in Figure 5.21 so that transitions *into* and *out of* intersections may both be used

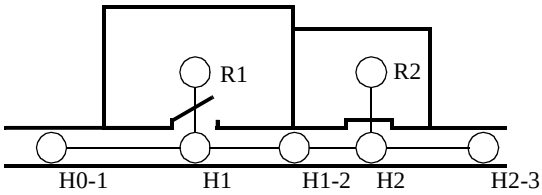


Fig 5.22 *A modification of the topological map to maximize information.*

for position updates. Figure 5.22 shows a modification of the topological map in which just this step has been taken. In this case, note that there are 7 nodes in contrast to 4.

In order to represent a specific belief state, Dervish associated with each topological node *n* a probability that the robot is at a physical position within the boundaries of *n*: $p(r_t = n)$.

As will become clear below, the probabilistic update used by Dervish was approximate, therefore technically one should refer to the resulting values as *likelihoods* rather than probabilities.

The perception update process for Dervish functions precisely as in Equation (5.21). Perceptual events are generated asynchronously, each time the feature extractor is able to recognize a large-scale feature (e.g. doorway, intersection) based on recent ultrasonic values. Each perceptual event consists of a percept-pair (a feature on one side of the robot or two features on both sides).

Table 5.1: Dervish's certainty matrix.

	Wall	*Closed Door*	*Open Door*	*Open Hallway*	*Foyer*
Nothing detected	*0.70*	*0.40*	*0.05*	*0.001*	*0.30*
Closed door detected	*0.30*	*0.60*	*0*	*0*	*0.05*
Open door detected	*0*	*0*	*0.90*	*0.10*	*0.15*
Open hallway detected	*0*	*0*	*0.001*	*0.90*	*0.50*

Given a specific percept pair *i*, Equation (5.21) enables the likelihood of each possible position *n* to be updated using the formula:

$$p(n|i) = p(i|n)p(n) \tag{5.23}$$

The value of $p(n)$ is already available from the current belief state of Dervish, and so the challenge lies in computing $p(i|n)$. The key simplification for Dervish is based upon the realization that, because the feature extraction system only extracts 4 total features and because a node contains (on a single side) one of 5 total features, every possible combination of node type and extracted feature can be represented in a 4 x 5 table.

Dervish's *certainty matrix* (show in Table 5.1) is just this lookup table. Dervish makes the simplifying assumption that the performance of the feature detector (i.e. the probability that it is correct) is only a function of the feature extracted and the actual feature in the node. With this assumption in hand, we can populate the *certainty matrix* with confidence esti-

mates for each possible pairing of perception and node type. For each of the five world features that the robot can encounter (wall, closed door, open door, open hallway and foyer) this matrix assigns a likelihood for each of the three one-sided percepts that the sensory system can issue. In addition, this matrix assigns a likelihood that the sensory system will fail to issue a perceptual event altogether (*nothing detected*).

For example, using the specific values in Table 5.1, if Dervish is next to an open hallway, the likelihood of mistakenly recognizing it as an open door is 0.10. This means that for any node n that is of type *Open Hallway* and for the sensor value i=*Open door*, $p(i|n) = 0.10$. Together with a specific topological map, the certainty matrix enables straightforward computation of $p(i|n)$ during the perception update process.

For Dervish's particular sensory suite and for any specific environment it intends to navigate, humans generate a specific certainty matrix that loosely represents its perceptual confidence, along with a global measure for the probability that any given door will be closed versus opened in the real world.

Recall that Dervish has no encoders and that perceptual events are triggered asynchronously by the feature extraction processes. Therefore, Dervish has no action update step as depicted by Equation (5.22). When the robot does detect a perceptual event, multiple perception update steps will need to be performed in order to update the likelihood of every possible robot position given Dervish's former belief state. This is because there is often a chance that the robot has traveled *multiple* topological nodes since its previous perceptual event (i.e. false negative errors). Formally, the perception update formula for Dervish is in reality a combination of the general form of action update and perception update. The likelihood of position n given perceptual event i is calculated as in Equation (5.22):

$$p(n_t|i_t) = \int p(n_t|n'_{t-i}, i_t)p(n'_{t-i})dn'_{t-i} \tag{5.24}$$

The value of $p(n'_{t-i})$ denotes the likelihood of Dervish being at position n' as represented by Dervish's former belief state. The temporal subscript *t-i* is used in lieu of *t-1* because for each possible position n' the discrete topological distance from n' to n can vary depending on the specific topological map. The calculation of $p(n_t|n'_{t-i}, i_t)$ is performed by multiplying the probability of generating perceptual event i at position n by the probability of having failed to generate perceptual events at all nodes between n' and n:

$$p(n_t|n'_{t-i}, i_t) = p(i_t, n_t) \cdot p(\varnothing, n_{t-1}) \cdot p(\varnothing, n_{t-2}) \cdot \ldots \cdot p(\varnothing, n_{t-i+1}) \tag{5.25}$$

For example (figure 5.23), suppose that the robot has only two nonzero nodes in its belief state, {1-2, 2-3}, with likelihoods associated with each possible position: p(1-2) = 1.0 and p(2-3) = 0.2. For simplicity assume the robot is facing East with certainty. Note that the likelihoods for nodes 1-2 and 2-3 do not sum to 1.0. These values are not formal probabilities, and so computational effort is minimized in Dervish by avoiding normalization altogether. Now suppose that a perceptual event is generated: the robot detects an open hallway

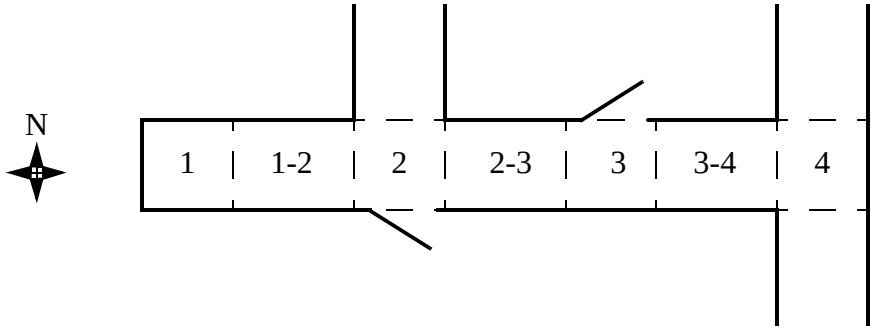


Fig 5.23 *A realistic indoor topological environment.*

on its left and an open door on its right simultaneously.

State 2-3 will progress potentially to states 3, 3-4 and 4. But states 3 and 3-4 can be eliminated because the likelihood of detecting an open door when there is only wall is zero. The likelihood of reaching state 4 is the product of the initial likelihood for state 2-3, 0.2, the likelihood of not detecting anything at node 3, (a), and the likelihood of detecting a hallway on the left and a door on the right at node 4, (b). Note that we assume the likelihood of detecting nothing at node 3-4 is 1.0 (a simplifying approximation).

(a) occurs only if Dervish fails to detect the door on its left at node 3 (either closed or open), [(0.6)(0.4) + (1-0.6)(0.05)], and correctly detects nothing on its right, 0.7.

(b) occurs if Dervish correctly identifies the open hallway on its left at node 4, 0.90, and mistakes the right hallway for an open door, 0.10.

The final formula, (0.2)[(0.6)(0.4)+(0.4)(0.05)](0.7)[(0.9)(0.1)], yields a likelihood of 0.003 for state 4. This is a partial result for $p(4)$ following from the prior belief state node 2-3.

Turning to the other node in Dervish's prior belief state, 1-2 will potentially progress to states 2, 2-3, 3, 3-4 and 4. Again, states 2-3, 3 and 3-4 can all be eliminated since the likelihood of detecting an open door when a wall is present is zero. The likelihood of state 2 is the product of the prior likelihood for state 1-2, (1.0), the likelihood of detecting the door on the right as an open door, [(0.6)(0) + (0.4)(0.9)], and the likelihood of correctly detecting an open hallway to the left, 0.9. The likelihood for being at state 2 is then (1.0)(0.4)(0.9)(0.9) = 0.3. In addition, 1-2 progresses to state 4 with a certainty factor of $4.3 \cdot 10^{-6}$, which is added to the certainty factor above to bring the total for state 4 to 0.00328. Dervish would therefore track the new belief state to be {2, 4}, assigning a very high likelihood to position 2 and a low likelihood to position 4.

Empirically, Dervish's map representation and localization system have proven to be sufficient for navigation of four indoor office environments: the artificial office environment created explicitly for the 1994 National Conference on Artificial Intelligence; the psychology department, the history department and the computer science department at Stanford University. All of these experiments were run while providing Dervish with no notion of the distance between adjacent nodes in its topological map. It is a demonstration of the power of probabilistic localization that, in spite of the tremendous lack of action and encoder information, the robot is able to navigate several real-world office buildings successfully.

One open question remains with respect to Dervish's localization system. Dervish was not just a localizer but also a navigator. As with all multiple hypothesis systems, one must ask the question, how does the robot decide how to move, given that it has multiple possible robot positions in its representation? The technique employed by Dervish is a most common technique in the mobile robotics field: plan the robot's actions by assuming that the robot's actual position is its most likely node in the belief state. Generally, the most likely position is a good measure of the robot's actual world position. However, this technique has shortcomings when the highest and second highest most likely positions have similar values. In the case of Dervish, it nonetheless goes with the highest likelihood position at all times, save at one critical juncture. The robot's goal is to enter a target room and remain there. Therefore, from the point of view of its goal, it is critical that it finish navigating only when the robot has strong confidence in being at the correct final location. In this particular case, Dervish's execution module refuses to enter a room if the gap between the most likely position and the second likeliest position is below a preset threshold. In such a case, Dervish will actively plan a path that causes it to move further down the hallway in an attempt to collect more sensor data and thereby increase the relative likelihood of one position in the belief state.

Although computationally unattractive, one can go further, imagining a planning system for robots such as Dervish for which one specifies a *goal belief state* rather than a goal position. The robot can then reason and plan in order to achieve a goal confidence level, thus explicitly taking into account not only robot position but also the measured likelihood of each position. An example of just such a procedure is the Sensory Uncertainty Field of Latombe [90], in which the robot must find a trajectory that reaches its goal while maximizing its localization confidence enroute.

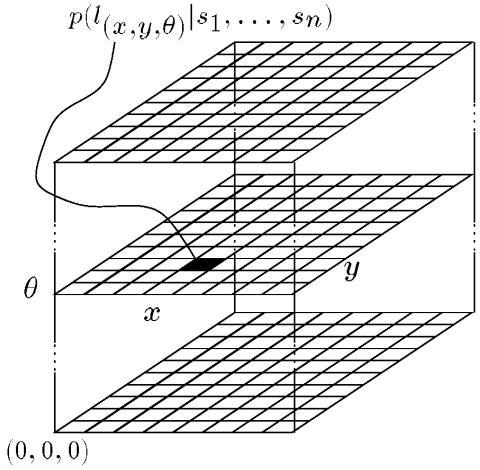


Fig 5.24 The belief state representation 3D array used by Rhino

5.6.2.3 Case Study II: 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 et al.* [42, 43] 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 24 sonars and 2 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 this 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 - 64cm in various instantiations.

Like Dervish, Rhino uses multiple hypothesis belief representation. In line with the far improved resolution of the environment representation, the belief state representation of Rhino consists of a 15 x 15 x 15 3D array representing the probability of 15^3 possible robot positions (see Figure 5.24). The resolution of the array is 15cm x 15cm x 1°. 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 environment representation in order for the overall system to function well.

Whereas Dervish made use 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:

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

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 3-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 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 $p(i|l)$ directly using a model of the robot's sensor behavior, its position l and the local environmental metric map around l.

The sensor model must calculate the probability of a specific perceptual measurement given that its likelihood is justified by known errors of the sonar or laser rangefinder sensors. Three key assumptions are used to construct this sensor model:

1 *If an object in the metric map is detected by a range sensor, the measurement error can be described with a distribution that has a mean at the correct reading.*

2 *There should always be a nonzero chance that a range sensor will read any measurement value, even if this measurement disagrees sharply with the environmental geometry.*

3 *In contrast to the generic error described in #2, there is a specific failure mode in ranging sensors whereby the signal is absorbed or coherently reflected, causing the sensor's range measurement to be maximal. Therefore, there is a local peak in the probability density distribution at the maximal reading of a range sensor.*

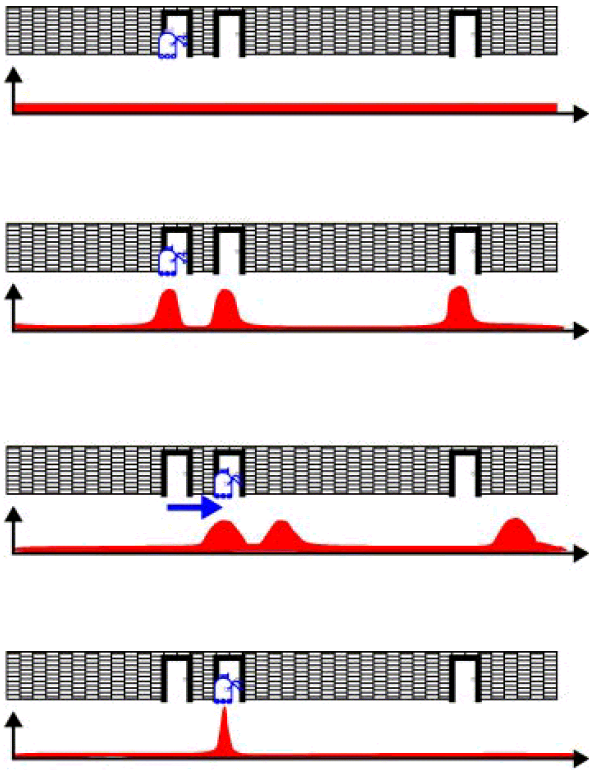


Fig 5.25 *Improving belief state by moving. Roland, we need to re-make this picture for copyright reasons probably.*

By validating these assumptions using empirical sonar trials in multiple environments, the research group has delivered to Rhino a conservative and powerful sensor model for its particular sensors.

Figure 5.25 provides a simple 1D example of the grid-based Markov localization algorithm. The robot begins with a flat probability density function for its possible location. In other words, it initially has no bias regarding position. As the robot encounters first one door and then a second door, the probability density function over possible positions becomes first multimodal and finally unimodal and sharply defined. The ability of a Markov localization system to *localize* the robot from an initially lost belief state is its key distinguishing feature.

The resulting robot localization system has been part of a navigation system that has demonstrated great success both at the University of Bonn and at a public museum in Bonn. This is a challenging application because of the dynamic nature of the environment, as the robot's sensors are frequently subject to occlusion due to humans gathering around the robot. Rhino's ability to function well in this setting is a demonstration of the power of the Markov localization approach

Reducing computational complexity: Randomized Sampling

A great many steps are taken in real-world implementations such as Rhino in order to effect computational gains. These are valuable because, with an exact cell decomposition representation and use of raw sensor values rather than abstraction to features, such a robot has a

massive computational effort associated with each perceptual update.

One class of techniques deserves mention because it can significantly reduce the computational overhead of techniques that employ fixed-cell decomposition representations. The basic idea, which we call *randomized sampling* is known alternatively as Particle filter algorithms, Condensation algorithms and Monte Carlo algorithms [ROLAND, reference the Thrun et al. paper on "Robust Monte Carlo" in my text file].

Irrespective of the specific technique, the basic algorithm is the same in all these cases. Instead of representing *every* possibe robot position by representing the complete and correct belief state, an approximate belief state is constructed by representing only a *subset* of the complete set of possible locations that should be considered.

For example, consider a robot with a complete belief state of 10,000 possible locations at time *t.* Instead of tracking and updating all 10,000 possible locations based on a new sensor measurement, the robot can select only 10% of the stored locations and update only those locations. By weighting this sampling process with the probability values of the locations, one can bias the system to generate more samples at local peaks in the probability density function. So, the resulting 1,000 locations will be concentrated primarily at the highest probability locations. This biasing is desirable, but only to a point.

We also wish to ensure that *some* less likely locations are tracked, as otherwise, if the robot does indeed receive unlikely sensor measurements, it will fail to localize. This *randomization* of the sampling process can be performed by adding additional samples from a flat distribution for example. Further enhancements of these randomized methods enable the number of statistical samples to be varied on-the-fly, based for instance on the ongoing localization confidence of the system. This further reduces the number of samples required on average while guaranteeing that a large number of samples will be used when necessary [ROLAND, reference the Fox NIPS article on KLD-Sampling Adaptive Particle Filters]

These sampling techniques have resulted in robots that function indistinguishably as compared to their full belief state set ancestors, yet use computationally a fraction of the resources. Of course, such sampling has a penalty: completeness. The probabilistically complete nature of Markov localization is violated by these sampling approaches because the robot is failing to update *all* the nonzero probability locations, and thus there is a danger that the robot, due to an unlikely but correct sensor reading, could become truly lost. Of course, recovery from a lost state is feasible just as with all Markov localization techniques.

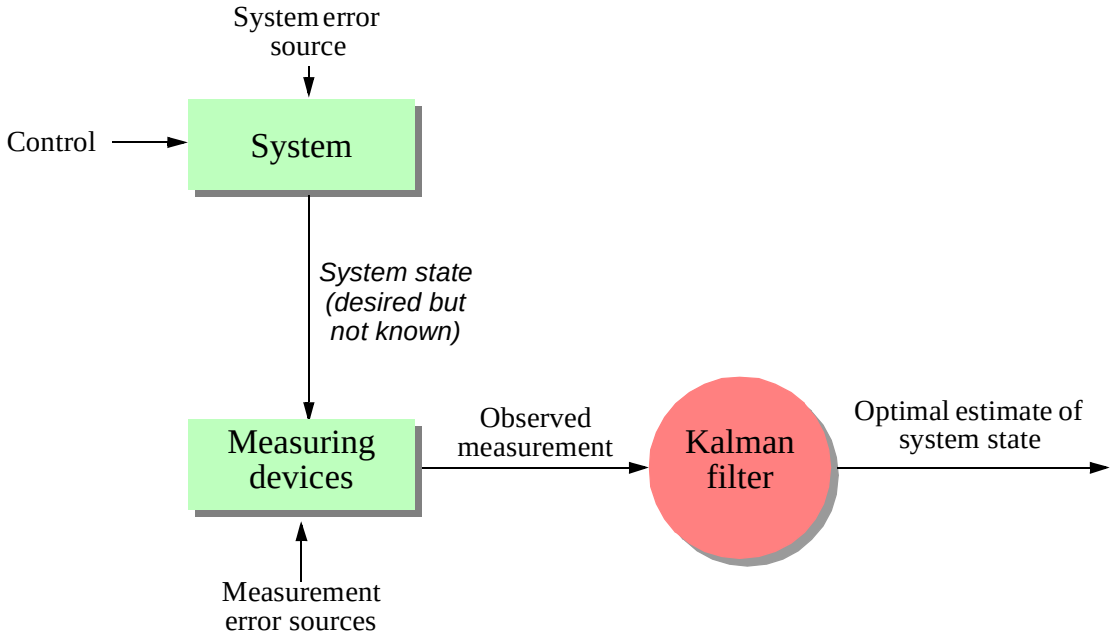


Fig 5.26 Typical Kalman filter application [46]

5.6.3 Kalman Filter Localization

The Markov localization model can represent any probability density function over robot position. This approach is very general but, due to its generality, inefficient. A successful alternative is to use a more compact representation of a specific class of probability densities. The Kalman filter does just this, and is an *optimal recursive data processing algorithm*. It incorporates all information, regardless of precision, to estimate the current value of the variable of interest. A comprehensive introduction can be found in [46] and a more detailed treatment is presented in [28].

Figure 5.26 depicts the a general scheme of Kalman filter estimation, where the system has a control signal and system error sources as inputs. A measuring device enables measuring some system states with errors. The Kalman filter is a mathematical mechanism for producing an optimal estimate of the system state based on the knowledge of the *system* and the *measuring device*, the description of the system noise and measurement errors and the uncertainty in the dynamics models. Thus the Kalman filter *fuses* sensor signals and system knowledge in an optimal way. Optimality depends on the criteria chosen to evaluate the performance and on the assumptions. Within the Kalman filter theory the system is assumed to be *linear* and *white* with *Gaussian* noise. As we have discussed earlier, the assumption of Gaussian error is invalid for our mobile robot applications but, nevertheless, the results are extremely useful. In other engineering disciplines, the Gaussian error assumption has in some cases been shown to be quite accurate [46].

We begin with a subsection that introduces Kalman filter theory, then we present an application of that theory to the problem of mobile robot localization. Finally, the third subsection will present a case study of a mobile robot that navigates indoor spaces by virtue of Kalman filter localization.

5.6.3.1 Introduction to Kalman Filter Theory

The basic Kalman filter method allows multiple measurements to be incorporated optimally into a single estimate of state. In demonstrating this, first we make the simplifying assumption that the state does not change (e.g. the robot does not move) between the acquisition of the first and second measurement. After presenting this static case, we can introduce *dynamic prediction* readily.

Static Estimation

Assume we have taken two measurements, one with an ultrasonic range sensor at time k and one with a more precise laser range sensor at time k+1. Based on each measurement we are able to estimate the robot's position. Such an estimate derived from the first sensor measurements is q_1 and the estimate of position based on the second measurement is q_2. Since we know that each measurement can be inaccurate, we wish to modulate these position estimates based on the measurement error expected from each sensor. Suppose that we use two variances σ_1^2 and σ_2^2 to predict the error associated with each measurement. We will, of course, assume a unimodal error distribution throughout the remainder of the Kalman filter approach, yielding the two robot position estimates:

$$\hat{q}_1 = q_1 \text{ with variance } \sigma_1^2 \tag{5.28}$$

$$\hat{q}_2 = q_2 \text{ with variance } \sigma_2^2. \tag{5.29}$$

So we have two measurements available to estimate the robots position. The question is, how do we *fuse* (combine) these data to get the best estimate $\hat{q}$ for the robot position?

We are assuming that there was no robot motion between time k and time k+1, and therefore we can directly apply the same weighted least square technique of Equation 4.61 in Section 4.3.1.1. Thus we write:

$$S = \sum_{i=1}^{n} w_i (\hat{q} - q_i)^2 \tag{5.30}$$

with w_i being the weight of measurement i. To find the minimum error we set the derivative of S equal to zero.

$$\frac{\partial S}{\partial \hat{q}} = \frac{\partial}{\partial \hat{q}} \sum_{i=1}^{n} w_i (\hat{q} - q_i)^2 = 2 \sum_{i=1}^{n} w_i (\hat{q} - q_i) = 0 \tag{5.31}$$

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

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

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 (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. This demonstrates that even poor measurements only increase the precision of an estimate (fig. 5.27), a result that we expect based on Information Theory.

Equation (5.35) can be rewritten as

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

or, in final form that is used in Kalman filter implementation

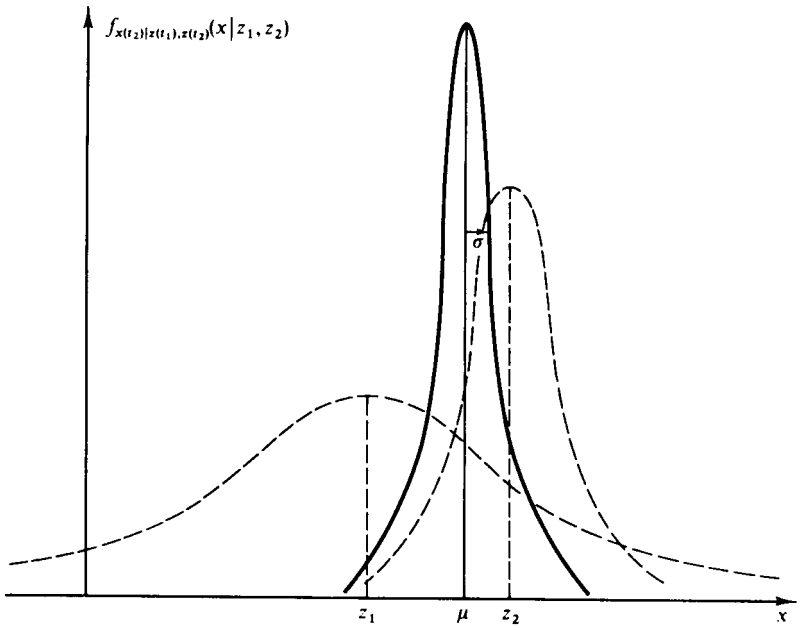


Fig 5.27 Fusing probability density of two estimates [46]

$$\hat{x}_{k+1} = \hat{x}_k + K_{k+1}(z_{k+1} - \hat{x}_k) \tag{5.38}$$

where

$$K_{k+1} = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} \quad ; \quad \sigma_k^2 = \sigma_1^2 \quad ; \quad \sigma_z^2 = \sigma_2^2 \tag{5.39}$$

Equation (5.38) tells us, that the best estimate $\hat{x}_{k+1}$ of the state x_{k+1} at time k+1 is equal to the best prediction of the value $\hat{x}_k$ before the new measurement z_{k+1} is taken, plus a correction term of an optimal weighting value times the difference between z_{k+1} and the best prediction $\hat{x}_{k+1}$ at time k+1. The updated variance of the state $\hat{x}_{k+1}$ is given using equation (5.36)

$$\sigma_{k+1}^2 = \sigma_k^2 - K_{k+1}\sigma_k^2 \tag{5.40}$$

Dynamic Estimation

Next, consider a robot that moves between successive sensor measurements. Suppose that the motion of the robot between times k and k+1 is described by the velocity u and the noise w which represents the uncertainty of the actual velocity:

$$\frac{dx}{dt} = u + w \tag{5.41}$$

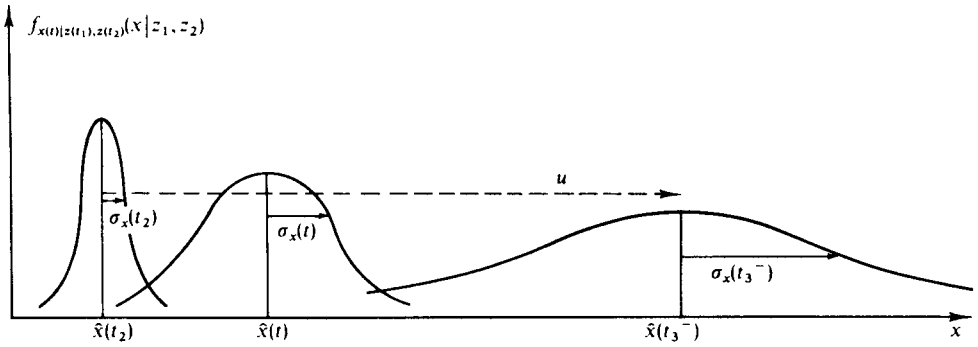


Fig 5.28 Propagation of probability density of a moving robot [46]

If we now start at time k, knowing the variance σ_k^2 of the robot position at this time and knowing the variance σ_w^2 of the motion, we obtain for the time k' just when the measurement is taken:

$$\hat{x}_{k'} = \hat{x}_k + u(t_{k+1} - t_k) \tag{5.42}$$

$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] \tag{5.43}$$

where:

$t_{k'} = t_{k+1}$

t_{k+1} and t_k are the time in seconds at k+1 and k respectively.

Thus $\hat{x}_{k'}$ is the optimal prediction of the robot's position just as the measurement is taken at time k+1. It describes the growth of position error until a new measurement is taken (fig. 5.28).

We can now rewrite equation (5.38) and (5.39) using equation (5.42) and (5.43).

$$\begin{aligned} {}_{k+1} &= \hat{x}_{k'} + K_{k+1}(z_{k+1} - \hat{x}_{k'}) \\ &= [\hat{x}_k + u(t_{k+1} - t_k)] + K_{k+1}[z_{k+1} - \hat{x}_k - u(t_{k+1} - t_k)] \end{aligned} \tag{5.44}$$

$$K_{k+1} = \frac{\sigma_{k'}^2}{\sigma_{k'}^2 + \sigma_z^2} = \frac{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]}{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] + \sigma_z^2} \tag{5.45}$$

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.

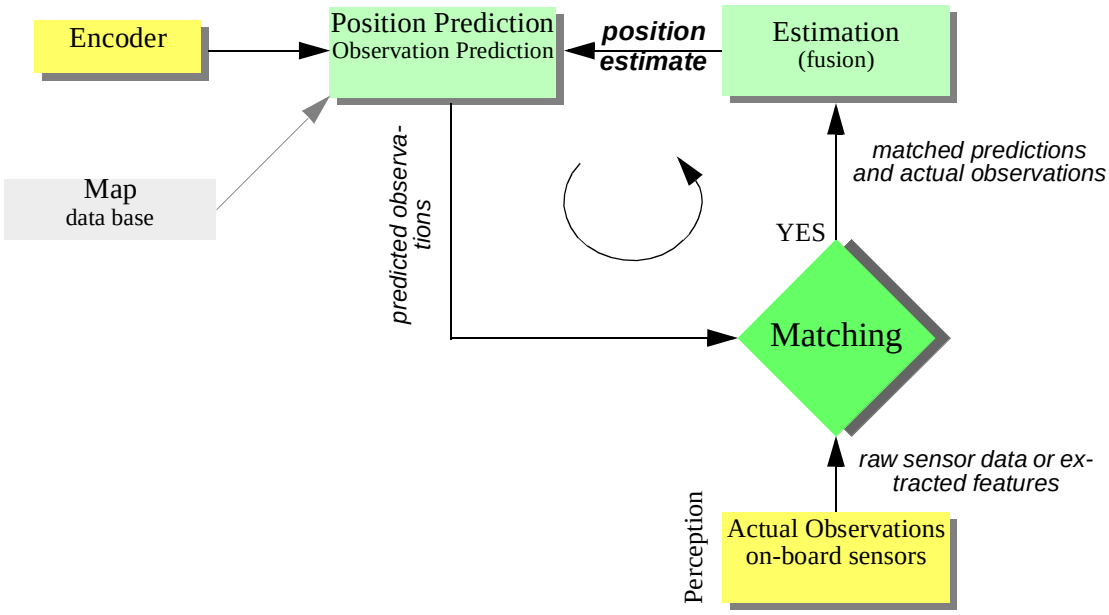


Fig 5.29 Schematic for Kalman filter mobile robot localization (see [9])

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, i.e. the robot's set of instantaneous sensor measurements, is used to update each possible robot position in the belief state individually using 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 multi-step process. The robot's total sensory input is treated, not as a monolithic whole, but as a set of extracted features that each relate to objects in the environment. Given a set of possible features, the Kalman filter is used to fuse the distance estimate from each feature to a matching object in the map. Instead of carrying out this matching process for many possible robot locations individually as in the Markov approach, the Kalman filter accomplishes the same probabilistic update by treating the whole, unimodal and Gaussian belief state at once. Figure 5.29 depicts the particular schematic for Kalman filter localization.

The first step is action update or *position prediction*, the straightforward application of a Gaussian error motion model to the robot's measured encoder travel. The robot then collects actual sensor data and extracts appropriate features (e.g. lines, doors, or even the value of a specific sensor) in the *observation* step. At the same time, based on its predicted position in the map, the robot generates a *measurement prediction* which identifies the features that the

robot expects to find and the positions of those features. In *matching* the robot identifies the best pairings between the features actually extracted during observation and the expected features due to measurement prediction. Finally, the Kalman filter can fuse the information provided by all of these matches in order to update the robot belief state in *estimation*.

In the following sub-sections these five steps are described in greater detail. The presentation is based on the work of Leonard and Durrant-Whyte [9 page 61-65].

1. Robot Position Prediction

The robot's position at time step k+1 is predicted based on its old location (at time step k) and its movement due to the control input $u(k)$:

$$\hat{p}(k+1|k) = f(\hat{p}(k|k), u(k)) \tag{5.46}$$

For the differential drive robot, $\hat{p}(k+1|k) = p'$ is derived in Equations (5.6) and (5.7) respectively.

Knowing the plant and error model, we can also compute the variance $\Sigma_p(k+1|k)$ associated with this prediction (see eq. (5.9) section 5.2.4):

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T \tag{5.47}$$

This allows us to predict the robot's position and its uncertainty after a movement specified by the control input $u(k)$. Note that the belief state is assumed to be Gaussian, and so we can characterize the believe state with just the two parameters $\hat{p}(k+1|k)$ and $\Sigma_p(k+1|k)$.

2. Observation

The second step it to obtain sensor measurements $Z(k+1)$ from the robot at time k+1. In this presentation, we assume that the observation is the result of a feature extraction process executed on the raw sensor data. Therefore, the observation consists of a set n_0 of single observations $z_j(k+1)$ extracted from various sensors. Formally each single observation can represent an extracted features such as a line or door, or even a single, raw sensor value.

The parameters of the features are usually specified in the sensor frame and therefore in a local reference frame of the robot. However, for matching we need to represent the observations and measurement predictions in the same frame $\{S\}$. In our presentation we will transform the measurement predictions from the global coordinate frame to the sensor frame $\{S\}$. This transformation is specified in the function h_i discussed in the next paragraph.

3. Measurement Prediction

We use the predicted robot position $\hat{p}(k+1|k)$ and the map $M(k)$ to generate multiple predicted feature observations z_t. Each predicted feature has its position transformed into the sensor frame:

$$\hat{z}_i(k+1) = h_i(z_t, \hat{p}(k+1|k)) \tag{5.48}$$

We can define the measurement prediction as the set containing all n_t predicted feature observations:

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

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 4.2.3 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:

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

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 hypothesis 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 [28] 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

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

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 one-dimensional 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:

$$\begin{aligned}(k+1|k+1) &= \hat{p}(k+1|k) + K(k+1) \cdot \ v(k+1) \\ &= \hat{p}(k+1|k) + K(k+1) \cdot \ [z_j(k+1) - h_i(z_t, \ \hat{p}(k+1|k))] \\ &= \hat{p}(k+1|k) + K(k+1) \cdot \ [z_j(k+1) - z_t]\end{aligned} \tag{5.57}$$

corresponding to equation (5.44).

5.6.3.3 Case study: Kalman filter localization with line feature extraction

The Pygmalion robot at EPFL is a differential drive robot that uses a laser rangefinder as its primary sensor [39], [76]. In contrast to both Dervish and Rhino, the environmental representation of Pygmalion is continuous and abstract: the map consists of a set of infinite lines describing the environment. Pygmalion's belief state is, of course, represented as a Gaussian distribution since this robot uses the Kalman filter localization algorithm. The value of its mean position μ is represented to a high level of precision, enabling Pygmalion to localize with very high precision when desired. Below, we present details for Pygmalion's implementation of the five Kalman filter localization steps. For simplicity we assume that the sensor frame $\{S\}$ is equal to the robot frame $\{R\}$. If not specified all the vectors are represented in the world coordinate system $\{W\}$.

1. Robot Position Prediction

At the time increment k the robot is at position $p(k) = \begin{bmatrix} x(k) & y(k) & \theta(k) \end{bmatrix}^T$ and its best position estimate is $\hat{p}(k|k)$. The control input $u(k)$ drives the robot to the position $p(k+1)$ (fig. 5.30).

The robot position prediction $\hat{p}(k+1)$ at the time increment k+1 can be computed from the previous estimate $\hat{p}(k|k)$ and the odometric integration of the movement. For the differential drive that Pygmalion has we can use the model (odometry) developed in section 5.2.4:

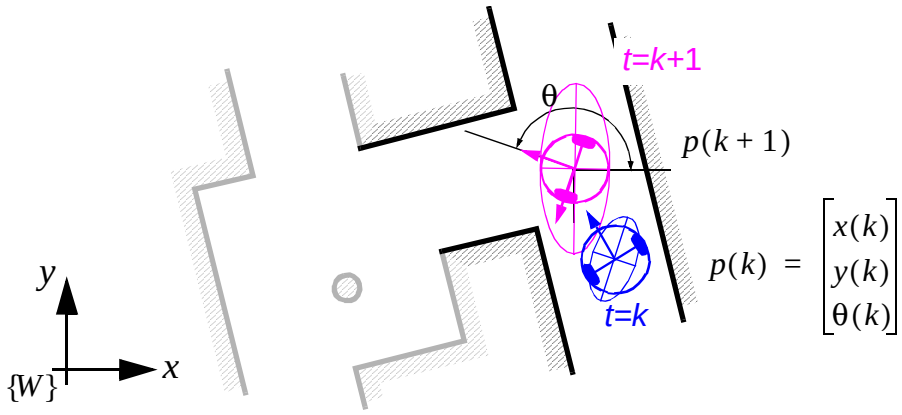


Fig 5.30 *Prediction of the robots position (magenta) based on its former position (blue) and the executed movement. The ellipses drawn around the robot positions represent the uncertainties in the x,y direction (e.g. 3σ). The uncertainty of the orientation θ is not represented in the picture.*

$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \hat{p}(k|k) + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.58}$$

with the updated covariance matrix

$$(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u \tag{5.59}$$

where

$$\Sigma_u = cov(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.60}$$

2. Observation

For line based localization, each single observation (i.e. a line feature) is extracted from the raw laser rangefinder data and consists of the two line parameters $\beta_{0,j}$, $\beta_{1,j}$ or α_j, r_j (fig. 4.36) respectively. For a rotating laser rangefinder, a representation in the polar coordinate frame is more appropriate and so we use this coordinate frame here:

$$z_j(k+1) = {}^R\begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} \tag{5.61}$$

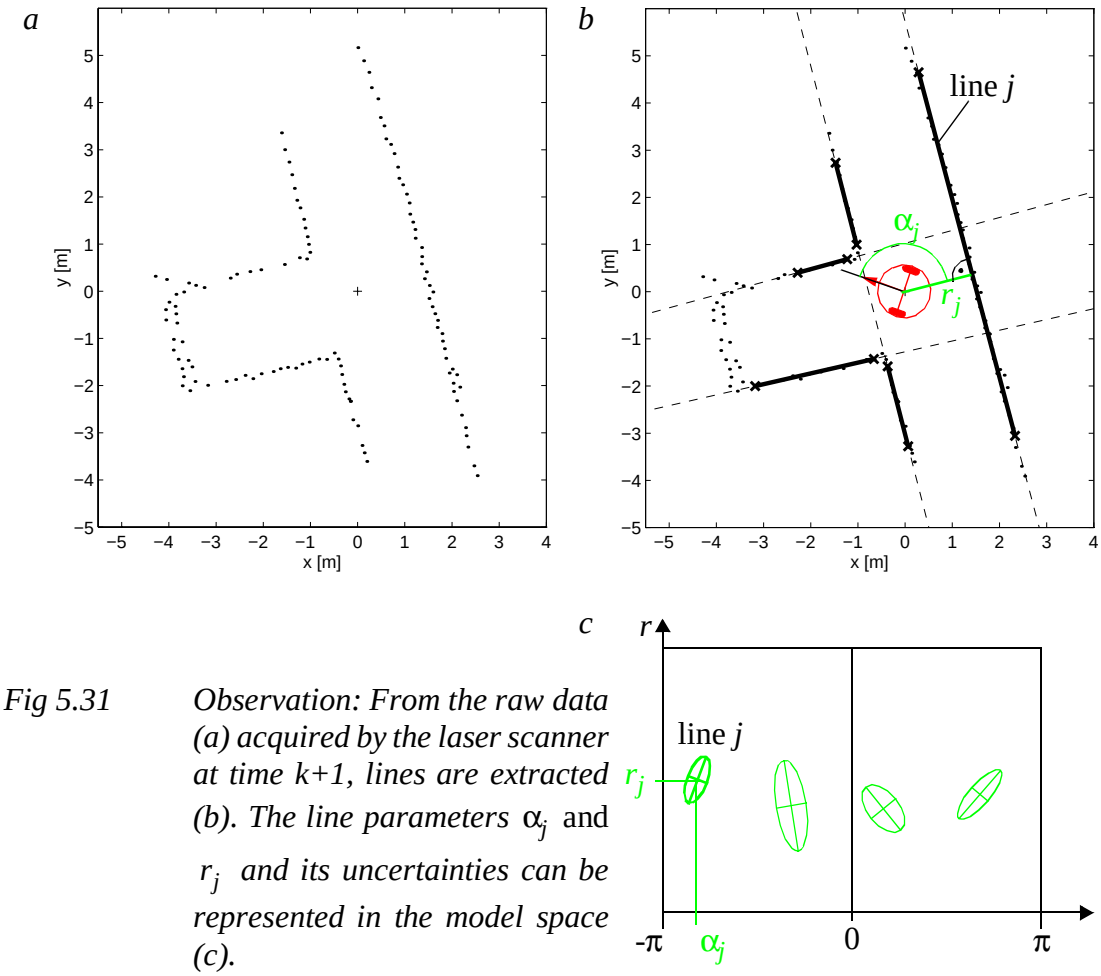


Fig 5.31 *Observation: From the raw data (a) acquired by the laser scanner at time k+1, lines are extracted (b). The line parameters α_j and r_j and its uncertainties can be represented in the model space (c).*

After acquiring the raw data at time k+1, lines and their uncertainties are extracted (fig. 5.31a/b). This leads to n_0 observed lines with $2n_0$ line parameters (5.31c) and a covariance matrix for each line that can be calculated from the uncertainties of all the measurement points contributing to each line as developed for line extraction in Section 4.3.1.1:

$$\Sigma_{R,\,j} = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix}_j \tag{5.62}$$

3. Measurement Prediction

Base on the stored map and the predicted robot position $\hat{p}(k|k)$, the measurement predictions of expected features $_{t,\,i}$ are generated (fig. 5.32). To reduce the required calculation power, there is often an additional step that first selects the possible features, in this case lines, from the whole set of features in the map. These lines are stored in the map and specified in the world coordinate system $\{W\}$. Therefore they need to be transformed to the robot frame $\{R\}$:

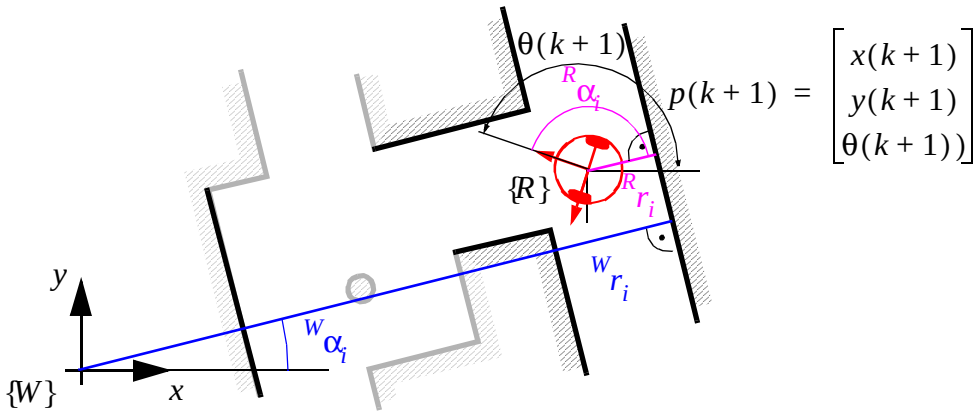


Fig 5.32 *Representation of the target position in the world coordinate frame* $\{W\}$ *and robot coordinate frame* $\{R\}$.

$$z_{t,\,i} = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{W} \rightarrow {}^{R}z_{t,\,i} = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{R} \tag{5.63}$$

According to figure (5.32), the transformation is given by

$$ {}_{i}(k+1) = \begin{bmatrix} \alpha_{t,\,i} \\ r_{t,\,i} \end{bmatrix}^{R} = h_i(z_{t,\,i},\ \hat{p}(k+1|k)) $$

$$ = \begin{bmatrix} {}^{W}\alpha_{t,\,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,\,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,\,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,\,i})) \end{bmatrix} \tag{5.64}$$

and its Jacobian ∇h_i by

$$\nabla h_i = \begin{bmatrix} \frac{\partial \alpha_{t,\,i}}{\partial \hat{x}} & \frac{\partial \alpha_{t,\,i}}{\partial \hat{y}} & \frac{\partial \alpha_{t,\,i}}{\partial \hat{\theta}} \\ \frac{\partial r_{t,\,i}}{\partial \hat{x}} & \frac{\partial r_{t,\,i}}{\partial \hat{y}} & \frac{\partial r_{t,\,i}}{\partial \hat{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos{}^{W}\alpha_{t,\,i} & -\sin{}^{W}\alpha_{t,\,i} & 0 \end{bmatrix} \tag{5.65}$$

The measurement prediction results in predicted lines represented in the robot coordinate frame (fig. 5.33). They are uncertain, because the prediction of robot position is uncertain.

4. Matching

For matching, we must find correspondence (or a pairing) between predicted and observed features (fig. 5.34). In our case we take the Mahalanobis distance

$$ {}_{ij}^{T}(k+1) \cdot \ \Sigma_{IN,\,ij}^{-1}(k+1) \cdot \ v_{ij}(k+1) \leq g^2 \tag{5.66}$$

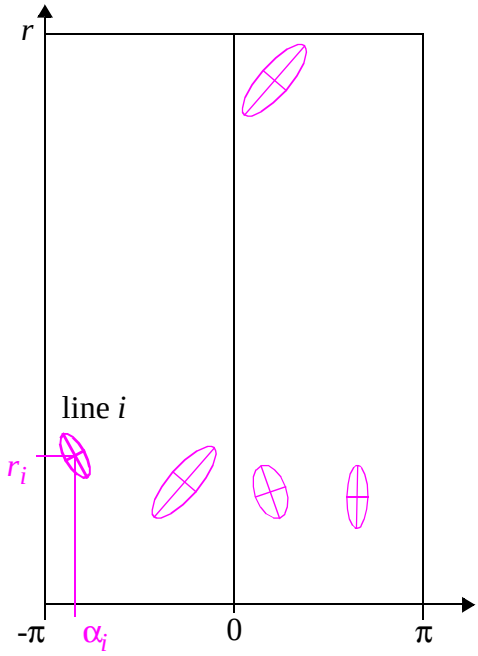


Fig 5.33 *Measurement predictions: Based on the map and the estimated robot position the targets (visible lines) are predicted. They are represented in the model space similar to the observations.*

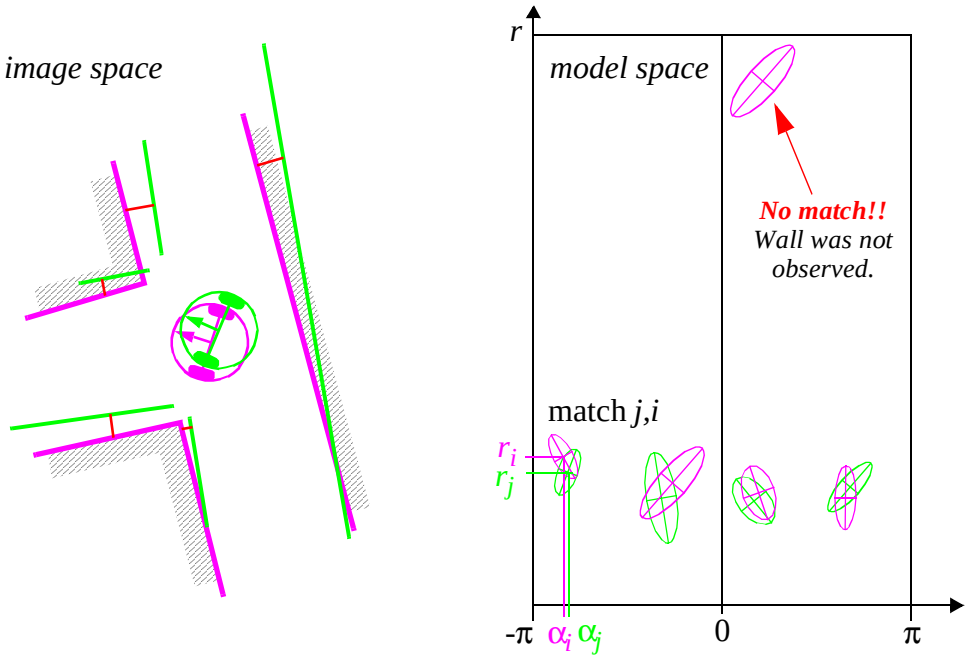


Fig 5.34 *Matching: The observations (green) and measurement prediction (magenta) are matched and the innovation and its uncertainties are calculated.*

with

$$v_{ij}(k+1) = [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))]$$
$$= \begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} - \begin{bmatrix} {}^W\alpha_{t,\,i} - {}^W\hat{\theta}(k+1|k) \\ {}^W r_{t,\,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,\,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,\,i})) \end{bmatrix} \quad (5.67)$$

$$\Sigma_{IN,\,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^T + \Sigma_{R,\,i}(k+1) \quad (5.68)$$

to enable finding the best matches while eliminating all other remaining observed and pre-

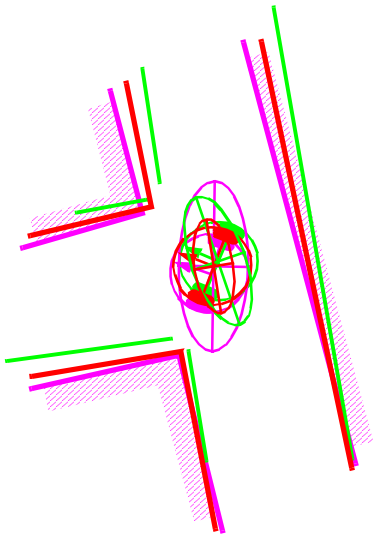


Fig 5.35 *Kalman filter estimation of the new robot position: By fusing the prediction of robot position (magenta) with the innovation gained by the measurements (green) we get the updated estimate $\hat{p}(k|k)$ of the robot position (red).*

dicted unmatched features.

5. Estimation

Applying the Kalman filter results in a final pose estimate corresponding to the weighted sum of (fig 5.35)

– *the pose estimates of each matched pairing of observed and predicted features*

– *robot position estimation based on odometry and observation positions*

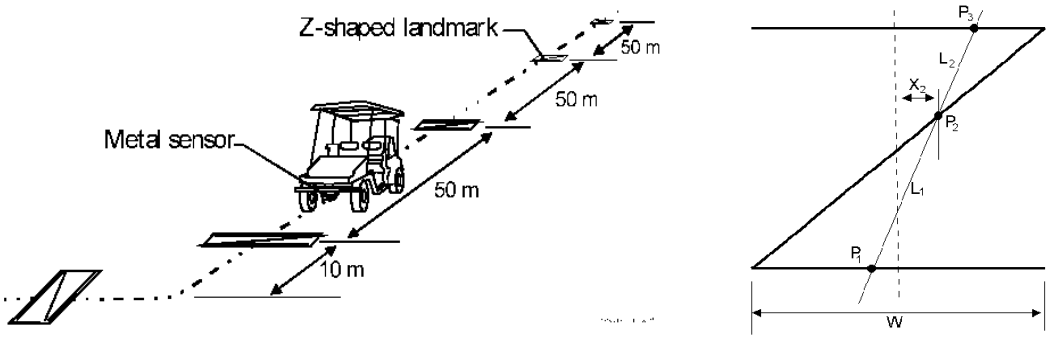


Fig 5.36 Z-shaped landmarks on the ground

5.7 Other Examples of Localization Systems

Markov localization and Kalman filter localization have been two extremely popular strategies for research mobile robot systems navigating indoor environments. They have strong formal bases and therefore well-defined behavior. But there are a large number of other localization techniques that have been used with varying degrees of success on commercial and research mobile robot platforms. We will not explore the space of all localization systems in detail. Refer to surveys such as [4] for such information.

There are, however, several categories of localization techniques that deserve mention. Not surprisingly, many implementations of these techniques in commercial robotics employ modifications of the robot's environment, something that the Markov localization and Kalman filter localization communities eschew. In the following sections, we briefly identify the general strategy incorporated by each category and reference example systems, including as appropriate those that modify the environment and those that function without environmental modification.

Landmark-based navigation

Landmarks are generally defined as passive objects in the environment that provide a high degree of localization accuracy when they are within the robot's field of view. Mobile robots that make use of landmarks for localization generally use artificial markers that have been placed by the robot's designers to make localization easy.

The control system for a landmark-based navigator consists of two discrete phases. When a landmark is in view, the robot localizes frequently and accurately, using action update and perception update to track its position without cumulative error. But when the robot is in no landmark "zone," then only action update occurs, and the robot accumulates position uncertainty until the next landmark enters the robot's field of view.

The robot is thus effectively *dead-reckoning* from landmark zone to landmark zone. This in turn means the robot must consult its map carefully, ensuring that each motion between landmarks is sufficiently short, given its motion model, that it will be able to localize successful upon reaching the next landmark.

Figure 5.36 shows one instantiation of landmark-based localization. The particular shape of the landmarks enables reliable and accurate pose estimation by the robot, which must travel

using *dead reckoning* between the landmarks.

One key advantage of the landmark-based navigation approach is that a strong formal theory has been developed for this general system architecture [113]. In this work, the authors have shown precise assumptions and conditions which, when satisfied, guarantee 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 meters 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 is usually required to cover a large factory area or research laboratory. For example, the Robotics Laboratory at Stanford made use of approximately 30 discrete landmarks, all affixed individually to the ceiling.

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 towards globally unique localization.

Figure (4.50) 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 for example Figure 4.51 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 5.37 is another example in which the robot's sensor values are transformed into an identifier of location. However, due to the limited infor-

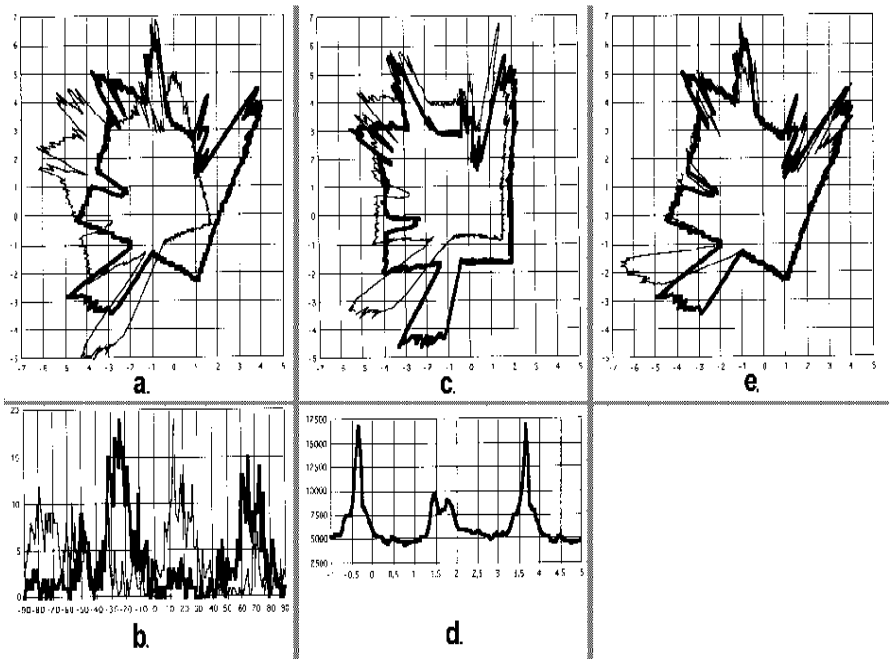


Fig 5.37 The angular histogram: Example

mation content of sonar ranging strikes, it is likely that two *places* in the robot's environment may have angular histograms that are too similar to be differentiated successfully.

One way of attempting to gather sufficient sonar information for global localization is to allow the robot time to gather a large amount of sonar data into a local evidence grid (i.e. occupancy grid) first, then match the local evidence grid with a global metric map of the environment. In [115] the researchers demonstrate such a system as able to localize on-the-fly even as significant changes are made to the environment, degrading the fidelity of the map. Most interesting is that the local evidence grid represents information well enough that it can be used to correct and update the map over time, thereby leading to a localization system that provides corrective feedback to the environment representation directly. This is similar in spirit to the idea of taking rejected observed features in the Kalman filter localization algorithm and using them to create new features in the map.

A most promising, new method for globally unique localization is called *Mosaic-based localization* [114]. This fascinating approach takes advantage of an environmental feature that is rarely used by mobile robots: fine-grained floor texture. This method succeeds primarily because of the recent ubiquity of very fast processors, very fast cameras and very large storage media.

The robot is fitted with a high-quality high-speed CCD camera pointed toward the floor, ideally situated between the robot's wheels and illuminated by a specialized light pattern off the camera axis to enhance floor texture. The robot begins by collecting images of the entire floor in the robot's workspace using this camera. Of course the memory requirements are significant, requiring a 10GB drive in order to store the complete image library of a 300 x 300 meter area.

Once the complete image mosaic is stored, the robot can travel any trajectory on the floor while tracking its own position without difficulty. Localization is performed by simply re-

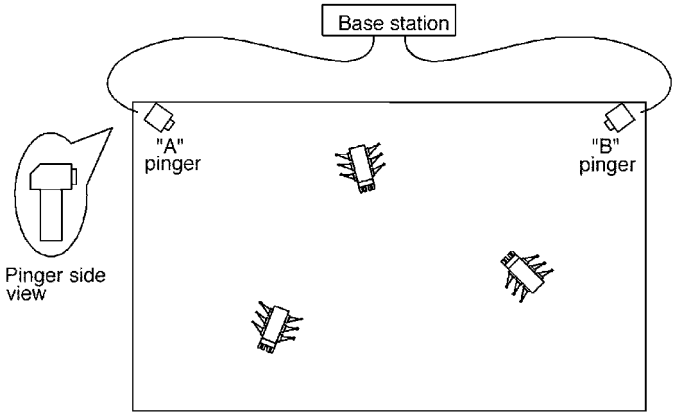


Fig 5.38 Active ultrasonic beacons.

cording one image, performing action update, then performing perception update by matching the image to the mosaic database using simple techniques based on image database matching. The resulting performance has been impressive: such a robot has been shown to localize repeatedly with 1mm precision while moving at 25 km/hr.

The key advantage of globally unique localization is that, when these systems function correctly, they greatly simplify robot navigation. The robot can move to any point and will always be assured of localizing by collecting a sensor scan.

But the main disadvantage of globally unique localization is that it is likely that this method will *never* offer a complete solution to the localization problem. There will always be cases where local sensory information is truly ambiguous and, therefore, globally unique localization using only current sensor information is unlikely to succeed. Humans often have excellent *local positioning systems*, particularly in non-repeating and well-known environments such as their homes. However, there are a number of environments in which such immediate localization is challenging even for humans: consider hedge mazes and large new office buildings with repeating halls that are identical. Indeed, the mosaic-based localization prototype described above encountered such a problem in its first implementation. The floor of the factory floor had been freshly painted and was thus devoid of sufficient micro-fractures to generate texture for correlation. Their solution was to modify the environment after all, painting random texture onto the factory floor.

Positioning Beacon systems

One of the most reliable solutions to the localization problem is to design and deploy an active beacon system specifically for the target environment. This is the preferred technique used by both industry and military applications as a way of ensuring the highest possible reliablility of localization. The GPS system can be considered as just such a system (see Section 4.1.5.1).

Figure 5.38 depicts one such beacon arrangement for a collection of robots. Just as with GPS, by designing a system whereby the robots localize passively while the beacons are active, any number of robots can simultaneously take advantage of a single beacon system. As

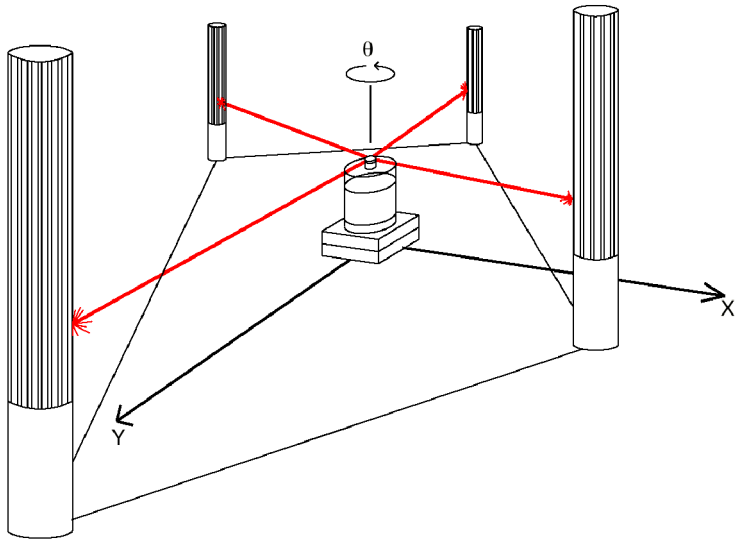


Fig 5.39 Passive optical beacons

with most beacon systems, the design depicted depends foremost upon geometric principles to effect localization. In this case the robots must know the positions of the two pinger units in the global coordinate frame in order to localize themselves to the global coordinate frame.

A popular type of beacon system in industrial robotic applications is depicted in Figure 5.39. In this case beacons are retroreflective markers that can be easily detected by a mobile robot based on their reflection of energy back to the robot. Given known positions for the optical retroreflectors, a mobile robot can identify its position whenever it has three such beacons in sight simultaneously. Of course, a robot with encoders can localize over time as well, and does not need to measure its angle to all three beacons at the same instant.

The advantage of such beacon-based systems is usually extremely high engineered reliability. By the same token, significant engineering usually surrounds the installation of such a system in a specific commercial setting. Therefore, moving the robot to a different factory floor will be both time-consuming and expensive. Usually, even changing the routes used by the robot will require serious re-engineering.

Route-based localization

Even more reliable than beacon-based systems are route-based localization strategies. In this case, the route of the robot is explicitly marked so that it can determine its position, not relative to some global coordinate frame, but relative to the specific path it is allowed to travel. There are many techniques for marking such a route and the subsequent intersections. In all cases, one is effectively creating a railway system, except that the railway system is somewhat more flexible and certainly more human-friendly than a physical rail. For example, high uv-reflective, optically transparent paint can mark the route such that only the robot, using a specialized sensor, easily detects it. Alternatively, a guide wire buried underneath the hall can be detected using inductive coils located on the robot chassis.

In all such cases, the robot localization problem is effectively trivialized by forcing the robot to always follow a prescribed path. To be fair, there are new industrial *unmanned guided*

vehicles that do deviate briefly from their route in order to avoid obstacles. Nevertheless, the cost of this extreme reliability is obvious: the robot is much more inflexible give such localization means, and therefore any change to the robot's behavior requires significant engineering and time.

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 is not so, 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 environment map. Of course a mobile robot's sensors have only 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 a 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 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 sub-section [75].

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 5.40 shows a general schematic incorporating map building and maintenance into the standard localization loop depicted by Figure (5.29) during discussion of Kalman filter localization [9]. The added arcs represent the additional flow of information that occurs when there is an imperfect match between observations and measurement predictions.

Unexpected observations will affect the creation of new features in the map whereas unobserved measurement predictions will affect the removal of features from the map. As discussed earlier, each specific prediction or observation has an unknown exact value and so it

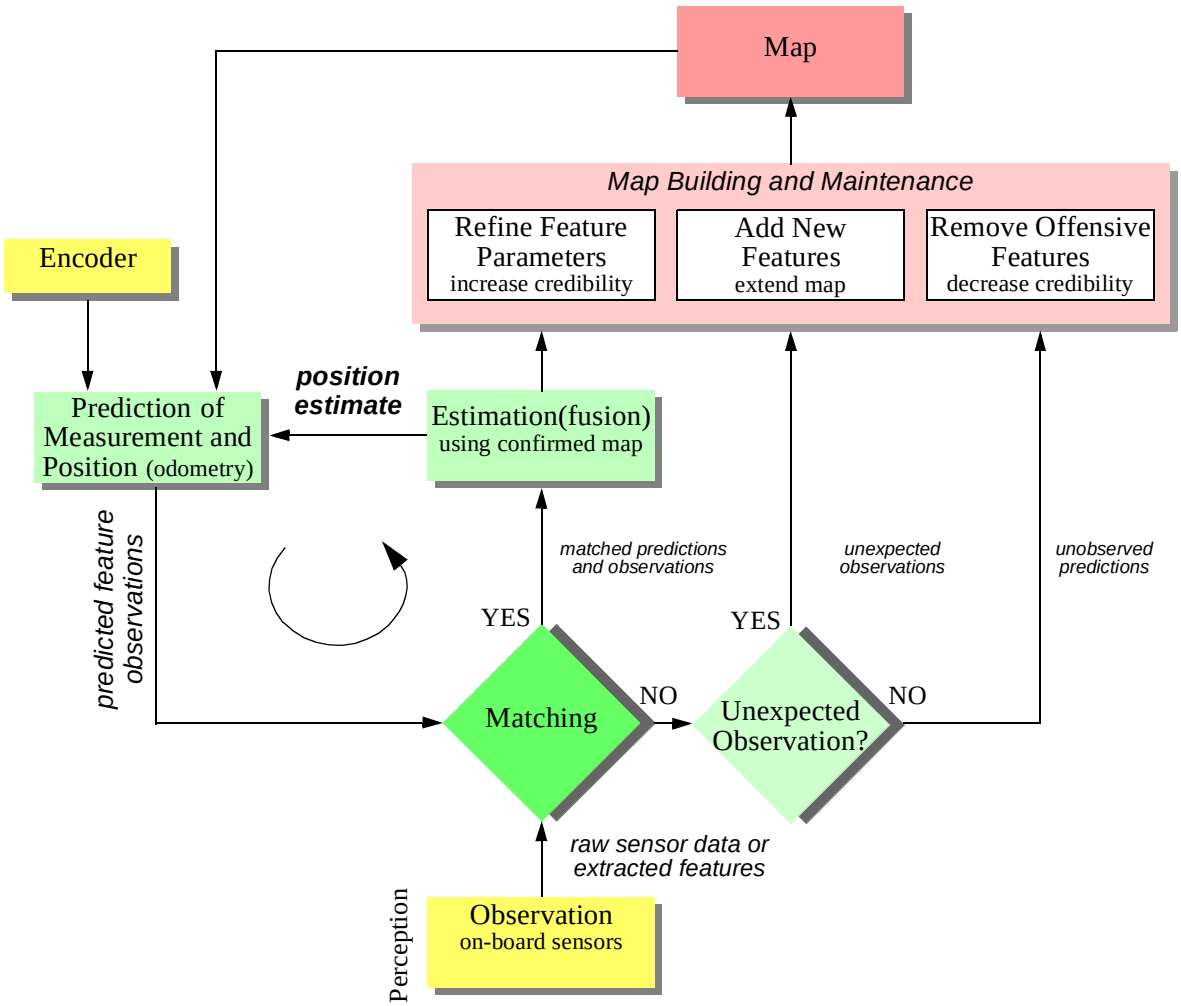


Fig 5.40 General schematic for concurrent localization and map building (see [9])

is represented by a distribution. The uncertainties of all of these quantities must be considered throughout this process.

The new type of map we are creating not only has features in it as did previous maps, but it also has varying degrees of probability that each feature is indeed part of the environment.

We represent this new map M with a set n of probabilistic feature locations $\hat{z}_t$, each with the covariance matrix Σ_t and an associated *credibility factor* c_t between 0 and 1 quantifying the belief in the existence of the feature in the environment (see Fig. (5.41)):

$$M = \{\hat{z}_t,\ \Sigma_t,\ c_t \mid (1 \le t \le n)\} \tag{5.69}$$

In contrast to the map used for Kalman filter localization previously, the map M is not assumed to be precisely known because it will be created by an uncertain robot over time. This is why the features $\hat{z}_t$ are described with associated covariance matrices Σ_t.

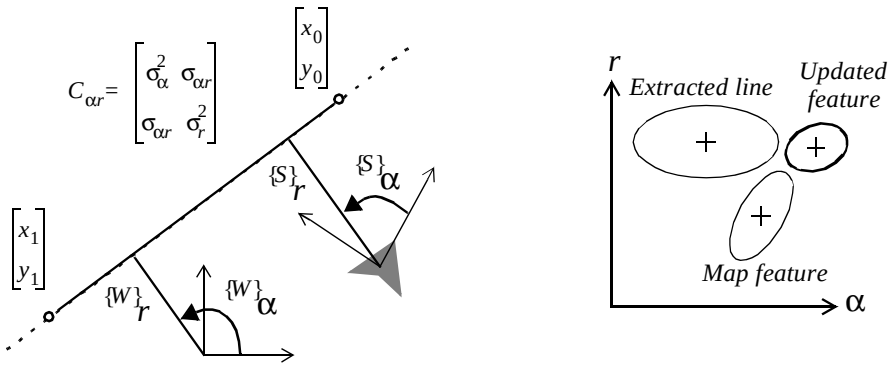


Fig 5.41 Uncertainties in the map.

Just as with Kalman filter localization, the matching step yields has three outcomes in regard to measurement predictions and observations: *matched prediction and observations*, *unexpected observations* and *unobserved predictions*. Localization, or the position update of the robot, proceeds as before. However, the map is also updated now, using all three outcomes and complete propagation of all the correllated uncertainties (see [9] for more details).

An interesting variable is the credibility factor c_t, which governs the likelihood that the mapped feature is indeed in the environment. How should the robot's failure to match observed features to a particular map feature reduce that map feature's credibility? And also, how should the robot's success at matching a mapped feature increase the chance that the mapped feature is "correct?" In [9] the following function is proposed for calculating credibility:

$$c_t(k) = 1 - e^{-\left(\frac{n_s}{a} - \frac{n_u}{b}\right)} \tag{5.70}$$

where a and b define the learning and forgetting rate and n_s and n_u are the number of matched and unobserved predictions up to time k, respectively. The update of the covariance matrix Σ_t can be done similarly to the position update seen in previous section. In map-building the feature positions and the robot's position are strongly correlated. This forces us to use a *stochastic map*, in which all cross-correlations must be updated in each cycle [73, 74, 75].

The stochastic map consists of a stacked system state vector:

$$X = \left[x_r(k)\ x_1(k)\ x_2(k)\ \ldots x_n(k)\right]^T \tag{5.71}$$

and a system state covariance matrix:

$$\Sigma = \begin{bmatrix} C_{rr} & C_{r1} & C_{r2} & \dots C_{rn} \\ C_{1r} & C_{11} & \dots & \dots C_{1n} \\ C_{2r} & \dots & \dots & \dots C_{2n} \\ \dots & \dots & \dots & \dots \dots \\ C_{nr} & C_{n1} & C_{n2} & \dots C_{nn} \end{bmatrix} \quad (5.72)$$

where the index r stands for the robot and the index $i = 1$ to n for the features in the map.

In contrast to localization based on an *a priori* accurate map, in the case of a stochastic map the cross-correlations must be maintained and updated as the robot is performing automatic map-building. During each localization cycle, the cross-correlations robot-to-feature and feature-to-robot are also updated. In short, this optimal approach requires every value in the map to depend on every other value, and therein lies the reason that such a complete solution to the automatic mapping problem is beyond the reach of even today's computational resources.

5.8.2 Other Mapping techniques

The mobile robotics research community has spent significant research effort on the problem of automatic mapping, and has demonstrating working systems in many environments without having solved the complete stochastic map problem described earlier. This field of mobile robotics research is extremely large, and this text will not present a comprehensive survey of the field. Instead, we present below two key considerations associated with automatic mapping, together with brief discussions of the approaches taken by several automatic mapping solutions to overcome these challenges.

5.8.2.1 Cyclic environments

Possibly the single hardest challenge for automatic mapping to be conquered is to correctly map cyclic environments. The problem is simple: given an environment that has one or more loops or cycles (e.g. four hallways that intersect to form a rectangle), create a globally consistent map for the whole environment.

This problem is hard because of the fundamental behavior of automatic mapping systems: the maps they create are not perfect. And, given any local imperfection, accumulating such imperfections over time can lead to arbitrarily large *global* errors between a map, at the macro level, and the real world, as shown in Figure (5.42). Such global error is usually irrelevant for mobile robot localization and navigation. After all, a warped map will still serve the robot perfectly well so long as the local error is bounded. However, an extremely large loop still eventually returns to the same spot, and the robot must be able to note this fact in its map. Therefore, global error does indeed matter in the case of cycles.

In some of the earliest work attempting to solve the cyclic environment problem, [116] used a purely topological representation of the environment, reasoning that the topological representation only captures the most abstract, most important features and avoids a great deal of

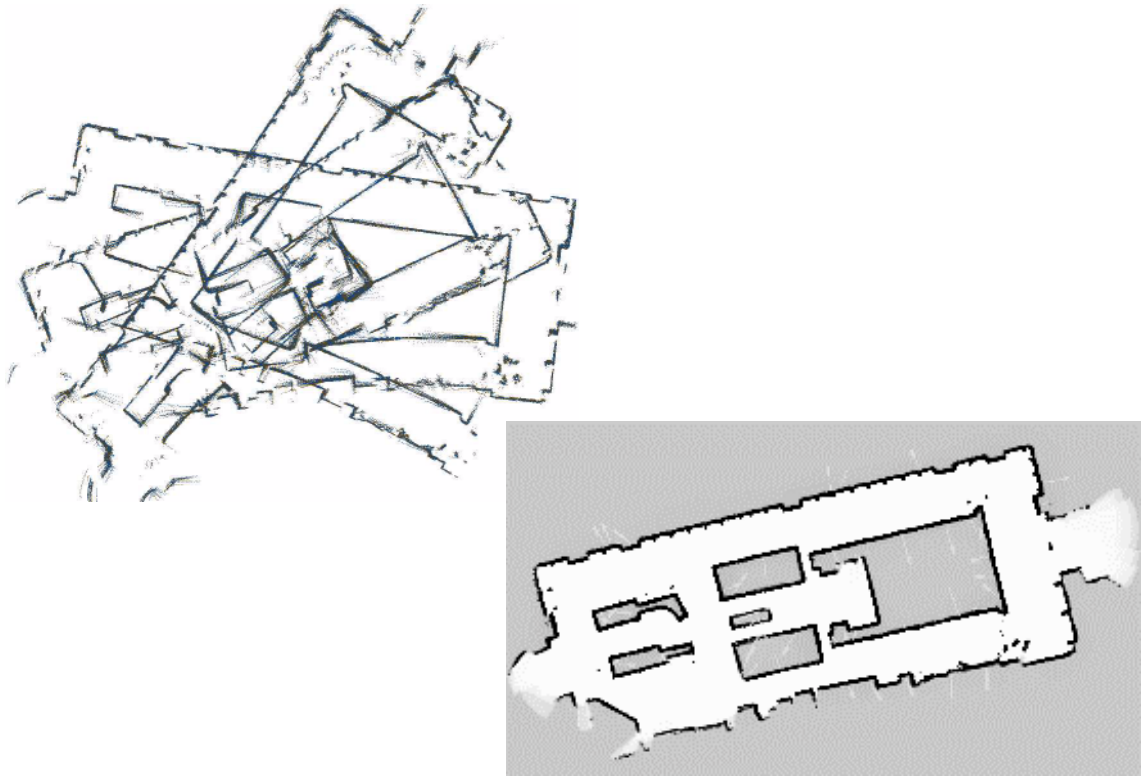


Fig 5.42 *Cyclic Environments: A naive, local mapping strategy with small local error leads to global maps that have a significant error, as demonstrated by this real-world run on the left. By applying topological correction, the grid map on the right is extracted [47].*

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 *sub-maps* [117, 118, 119]. Each *sub-map* is treated as a single sensor during the robot's position update. The advantage of this approach is two-fold. Because odometry is relatively accurate over small distances, the relative registration of features and raw sensor strikes in a local *sub-map* 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 *sub-maps*, explicitly identifying the

loops first at the topological level. In the case of [118] for example, the topological level loop is identified by a human who pushes a button at a known landmark position. In the case of [119] the topological level loop is determined by performing correspondence tests between *sub-maps*, postulating that two *sub-maps* 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 automonous mapping solutions but even 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 3:00 past midnight. However, in a great many cases this assumption is incorrect. In the case of wide-open spaces that are popular gathering places for humans, there is rapid change in the freespace and a vast majority of sensor strikes represent detection of the transient humans rather than fixed surfaces such as the perimeter wall. Another class of dynamic environments are spaces such as factory floors and warehouses, where the objects being stored redefine the topology of the pathways on a day-to-day basis as shipments are moved in and out.

In all such dynamic environments, an automatic mapping system should capture the *salient* objects detected by its sensors and, furthermore, the robot should have the flexibility to modify its map as the position of these salient objects changes. The subject of *continuous mapping*, or mapping of dynamic environments is to some degree a direct outgrowth of successful strategies for automatic mapping of unfamiliar environments. For example, in the case of stochastic mapping using the credibility factor c_t mechanism, the credibility equation can continue to provide feedback regarding the probability of existence of various mapped features after the initial map creation process is ostensibly complete. Thus, a mapping system can become a map-modifying system by simply continuing to operate. This is most effective, of course, if the mapping system is real-time and incremental. If map construction requires off-line global optimization, then the desire to make small-grained, incremental adjustments to the map is more difficult to satisfy.

Earlier we stated that a mapping system should capture only the *salient* objects detected by its sensors. One common argument for handling the detection of, for instance, humans in the environment is that mechanisms such as c_t can take care of all features that did not deserve to be mapped in the first place. For example, in [117] the authors develop a system based on a set of local occupany grids (called *evidence grids*) and a global occupancy grid.

Each time the robot's most recent local evidence grid is used to update a region of the global occupancy grid, extraneous occupied cells in the global occupancy grid are freed if the local occupancy grid detected no objects (with high confidence) at those same positions.

The general solution to the problem of detecting salient features, however, begs a solution to the perception problem in general. When a robot's sensor system can reliably detect the difference between a wall and a human, using for example a vision system, then the problem of mapping in dynamic environments will become significantly more straightforward.

We have discussed just two important considerations for automatic mapping. There is still a great deal of research activity focusing on the general map building and localization problem [9, 6, 47, 48, 49, 50, 75, 77]. However, there are few groups working on the general problem of probabilistic map building (i.e. stochastic maps) and, so far, a consistent and absolutely general solution has yet to be found. This field is certain to produce significant new results in the next several years, and as the perceptual power of robots improves we expect the payoff to be greatest here.