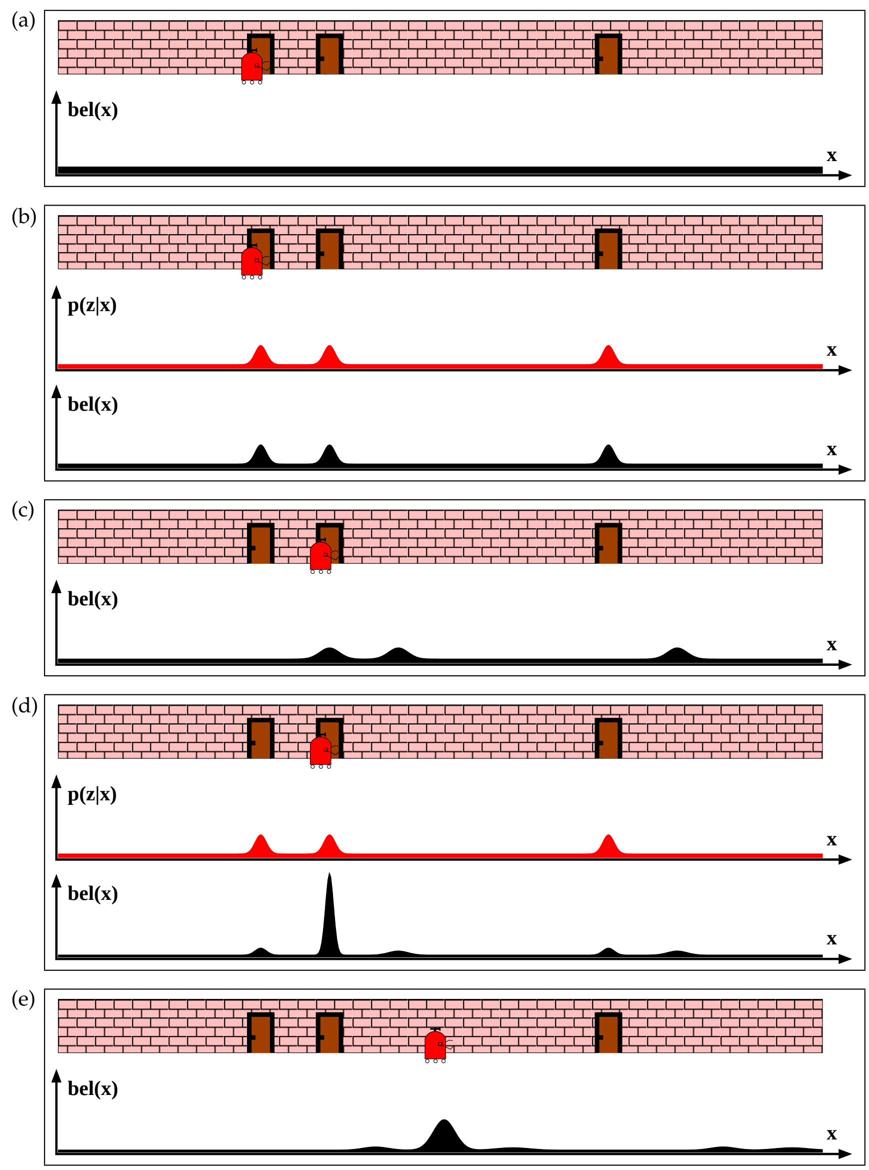
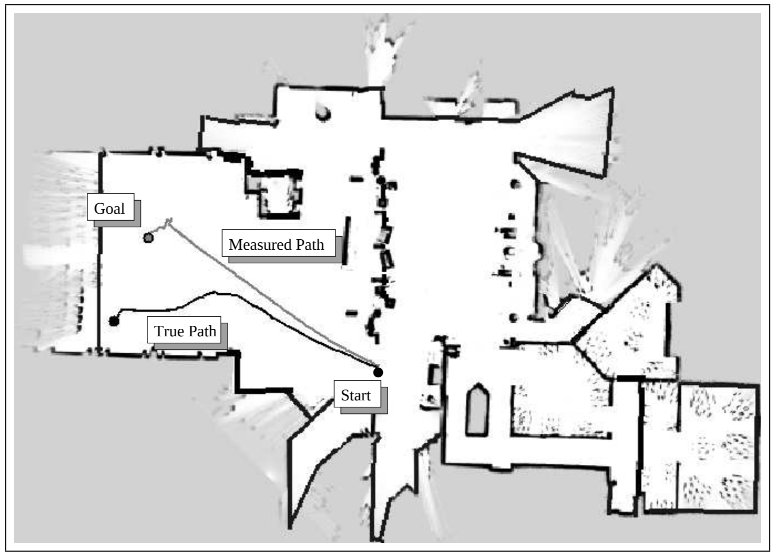


Figure 1.1 The basic idea of *Markov localization*: A mobile robot during global localization. Markov localization techniques will be investigated in Chapters **??** and **??**.

(a)



(b)

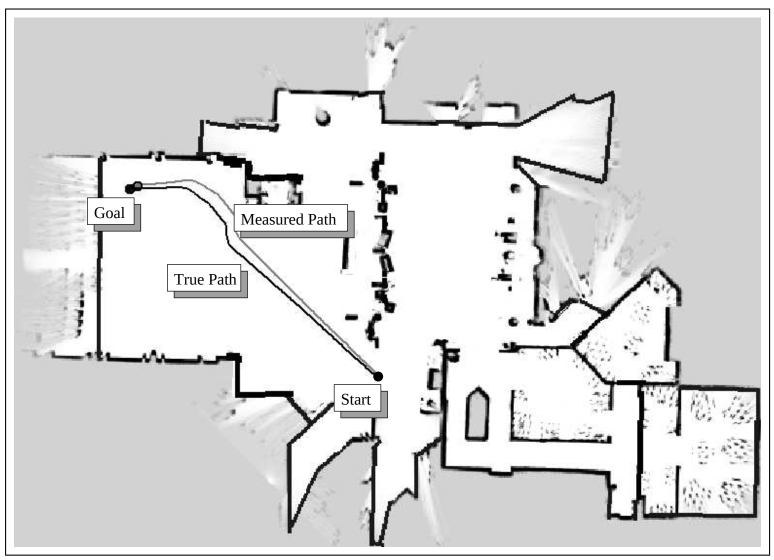


Figure 1.2 Top image: a robot navigating through open, featureless space may lose track of where it is. Bottom: This can be avoided by staying near known obstacles. These figures are results of an algorithm called *coastal navigation*, which will be discussed in Chapter **??**. Images courtesy of Nicholas Roy, MIT.

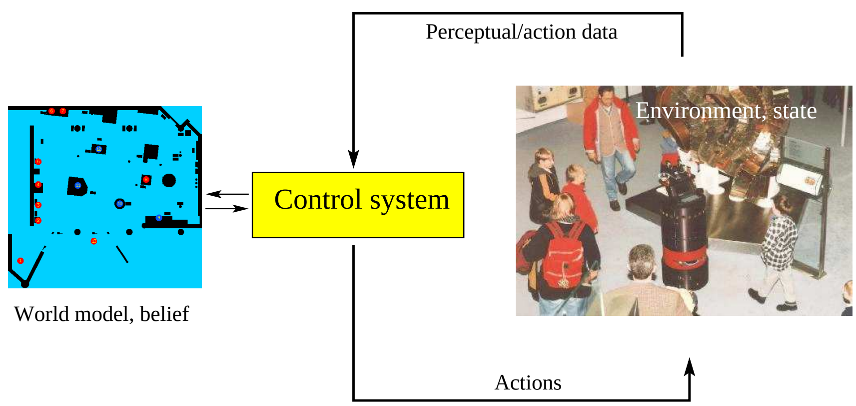


Figure 2.1 Robot environment interaction.

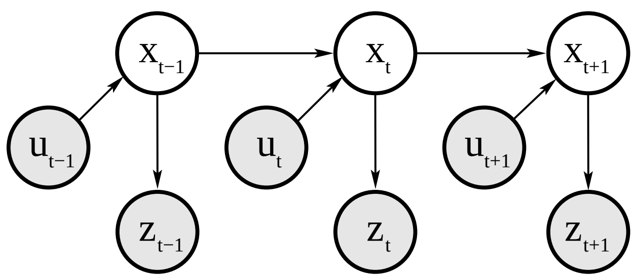


Figure 2.2 The dynamic Bayes network that characterizes the evolution of controls, states, and measurements.

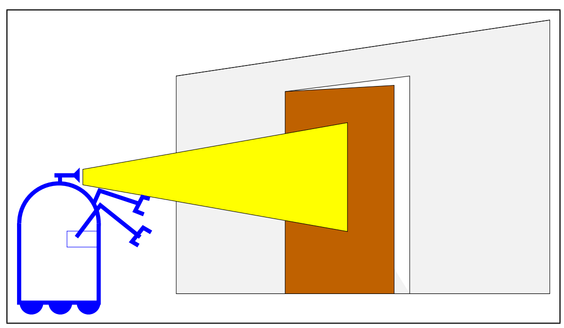


Figure 2.3 A mobile robot estimating the state of a door.

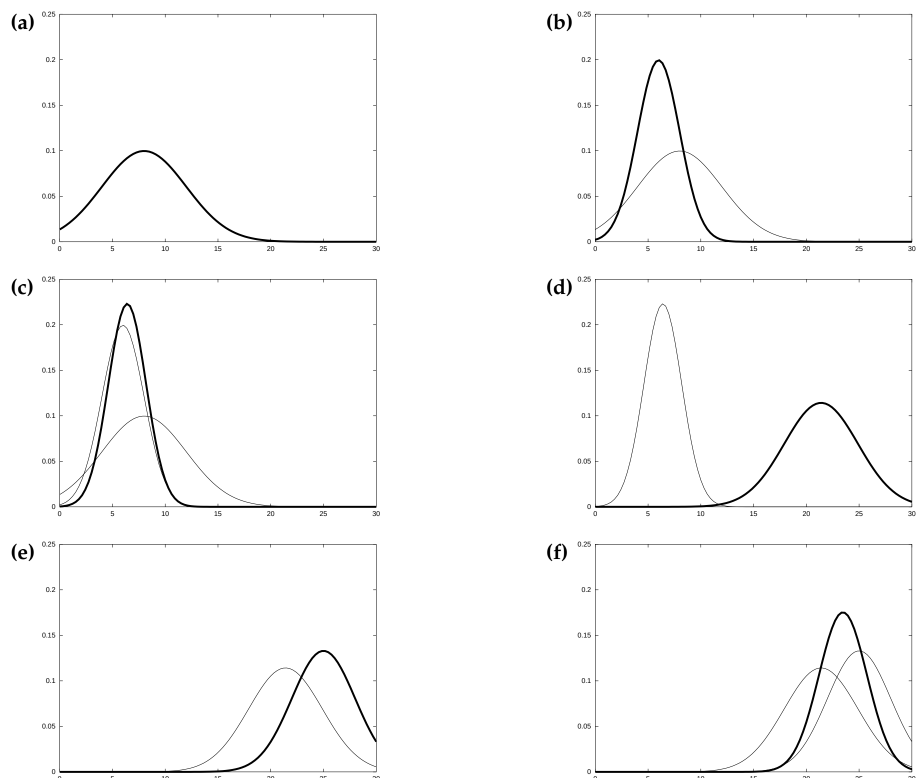


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement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

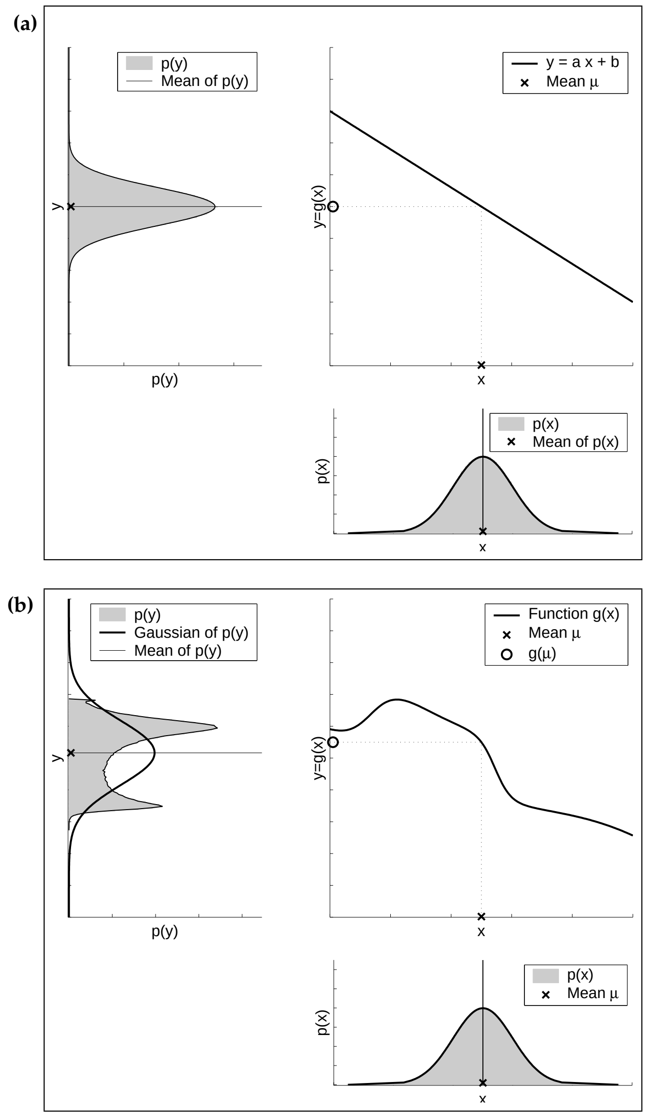


Figure 3.3 (a) Linear and (b) nonlinear transformation of a Gaussian random variable. The lower right plots show the density of the original random variable, X. This random variable is passed through the function displayed in the upper right graphs (the transformation of the mean is indicated by the dotted line). The density of the resulting random variable Y is plotted in the upper left graphs.

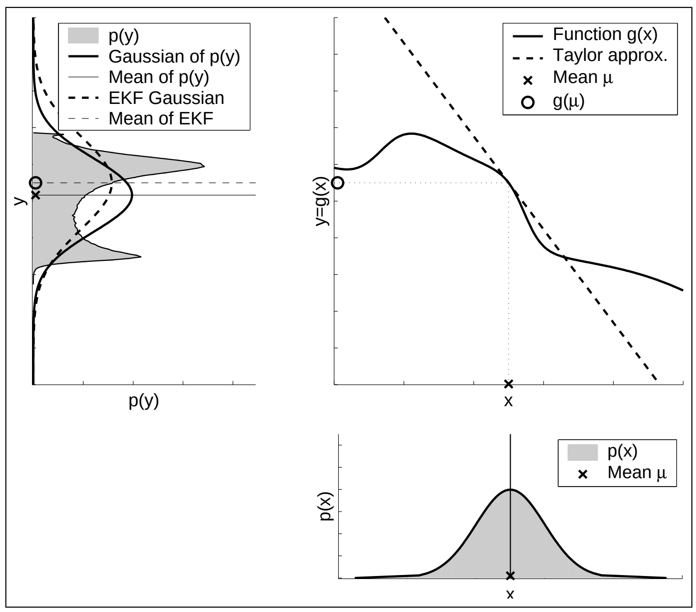


Figure 3.4 Illustration of linearization applied by the EKF. Instead of passing the Gaussian through the nonlinear function g, it is passed through a linear approximation of g. The linear function is tangent to g at the mean of the original Gaussian. The resulting Gaussian is shown as the dashed line in the upper left graph. The linearization incurs an approximation error, as indicated by the mismatch between the linearized Gaussian (dashed) and the Gaussian computed from the highly accurate Monte-Carlo estimate (solid).

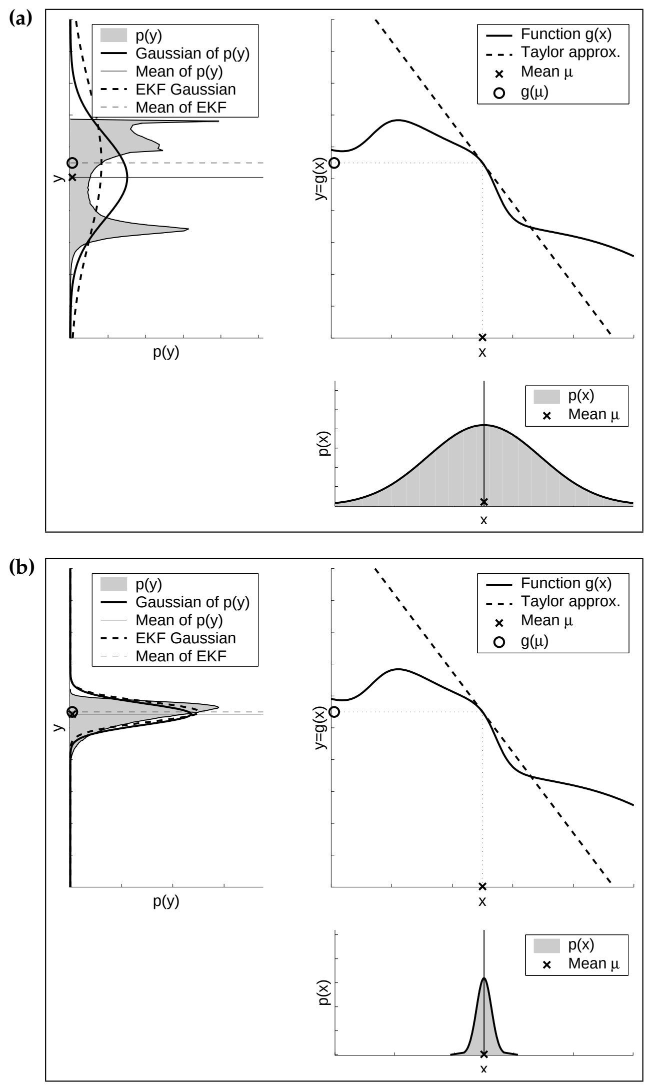


Figure 3.5 Dependency of approximation quality on uncertainty. Both Gaussians (lower right) have the same mean and are passed through the same nonlinear function (upper right). The higher uncertainty of the left Gaussian produces a more distorted density of the resulting random variable (gray area in upper left graph). The solid lines in the upper left graphs show the Gaussians extracted from these densities. The dashed lines represent the Gaussians generated by the linearization applied by the EKF.

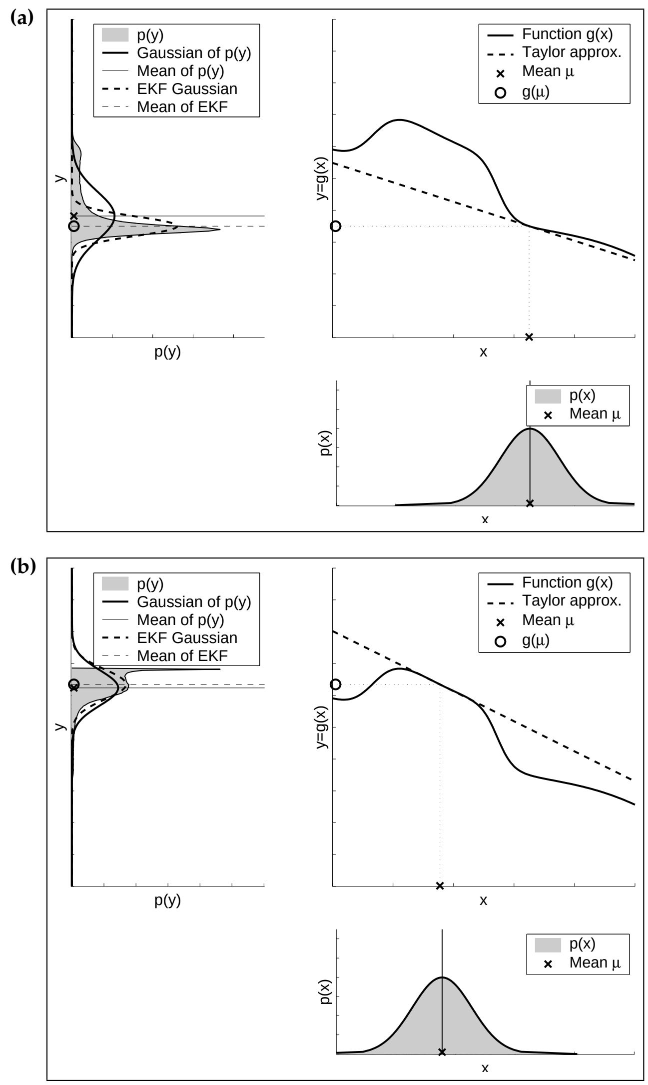


Figure 3.6 Dependence of the approximation quality on local nonlinearity of the function g. Both Gaussians (lower right in each of the two panels) have the same covariance and are passed through the same function (upper right). The linear approximation applied by the EKF is shown as the dashed lines in the upper right graphs. The solid lines in the upper left graphs show the Gaussians extracted from the highly accurate Monte-Carlo estimates. The dashed lines represent the Gaussians generated by the EKF linearization.

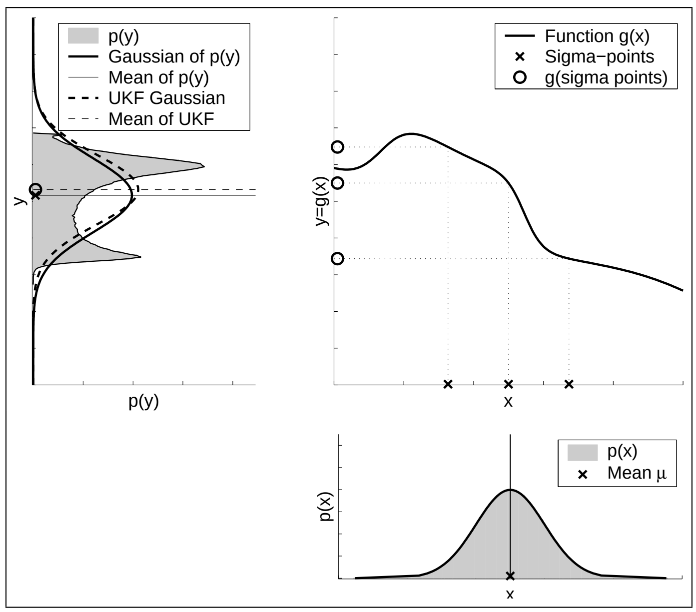


Figure 3.7 Illustration of linearization applied by the UKF. The filter first extracts $2n + 1$ weighted sigma points from the n-dimensional Gaussian ($n = 1$ in this example). These sigma points are passed through the nonlinear function g. The linearized Gaussian is then extracted from the mapped sigma points (small circles in the upper right plot). As for the EKF, the linearization incurs an approximation error, indicated by the mismatch between the linearized Gaussian (dashed) and the Gaussian computed from the highly accurate Monte-Carlo estimate (solid).

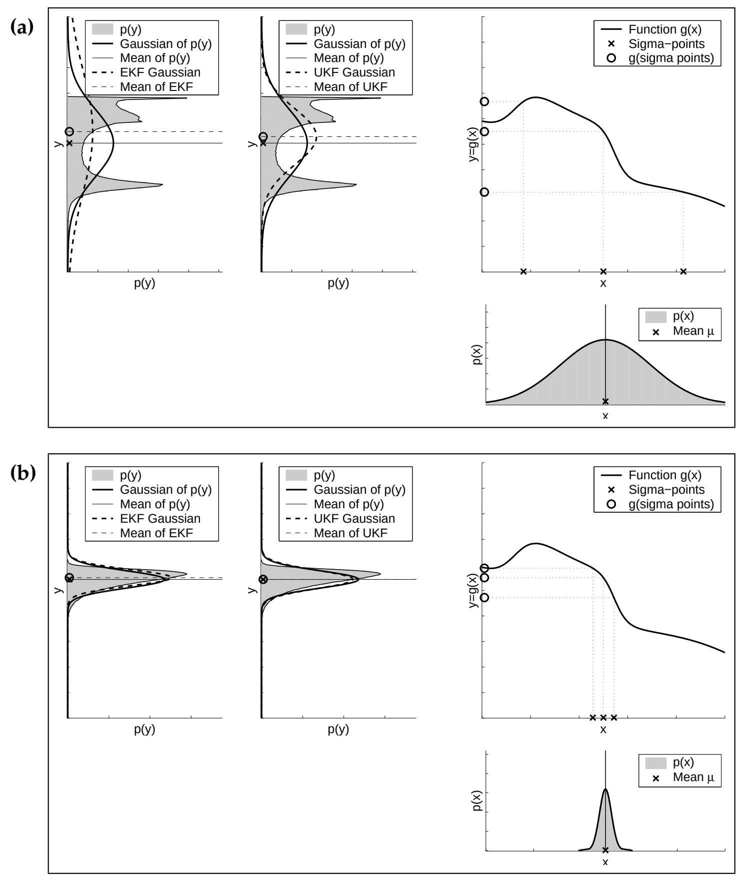


Figure 3.8 Linearization results for the UKF depending on the uncertainty of the original Gaussian. The results of the EKF linearization are also shown for comparison (c.f. Figure3.5). The unscented transform incurs smaller approximation errors, as can be seen by the stronger similarity between the dashed and the solid Gaussians.

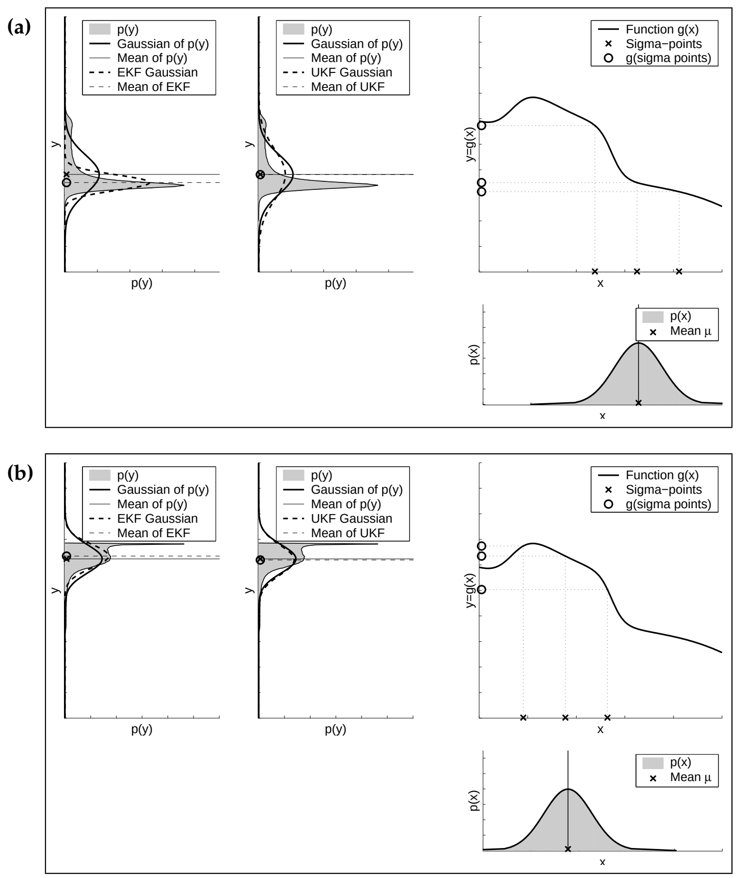


Figure 3.9 Linearization results for the UKF depending on the mean of the original Gaussian. The results of the EKF linearization are also shown for comparison (c.f. Figure3.6). The sigma point linearization incurs smaller approximation errors, as can be seen by the stronger similarity between the dashed and the solid Gaussians.

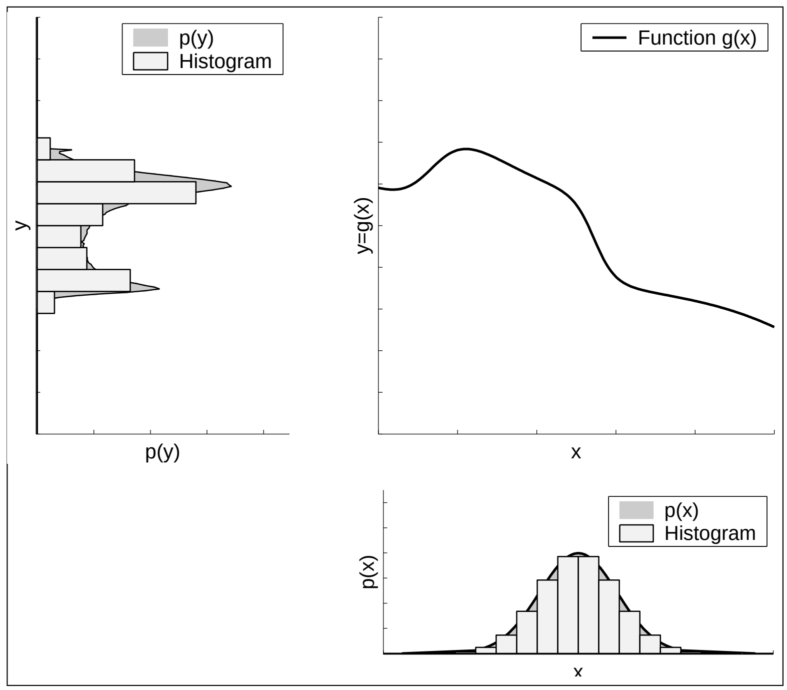


Figure 4.1 Histogram representation of a continuous random variable. The gray shaded area in the lower right plot shows the density of the continuous random variable, X. The histogram approximation of this density is overlaid in light-gray. The random variable is passed through the function displayed in the upper right graph. The density and the histogram approximation of the resulting random variable, Y, are plotted in the upper left graph. The histogram of the transformed random variable was computed by passing multiple points from each histogram bin of X through the nonlinear function.

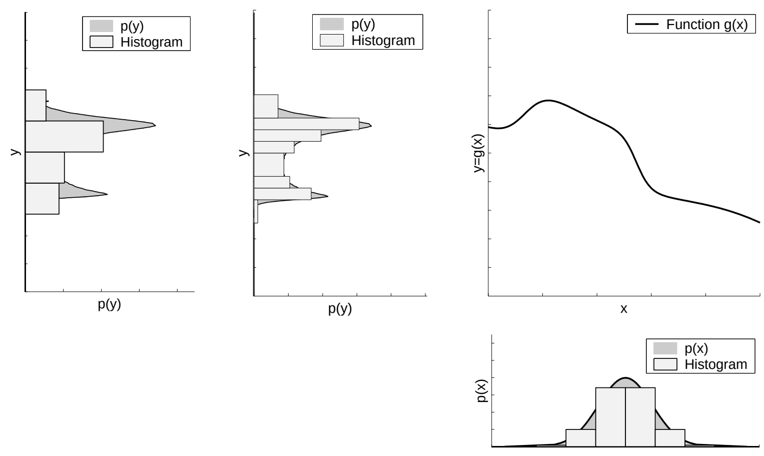


Figure 4.2 Dynamic vs. static decomposition. The upper left graph shows the static histogram approximation of the random variable Y, using 10 bins for covering the domain of Y (of which 6 are of nearly zero probability). The upper middle graph presents a tree representation of the same random variable, using the same number of bins.

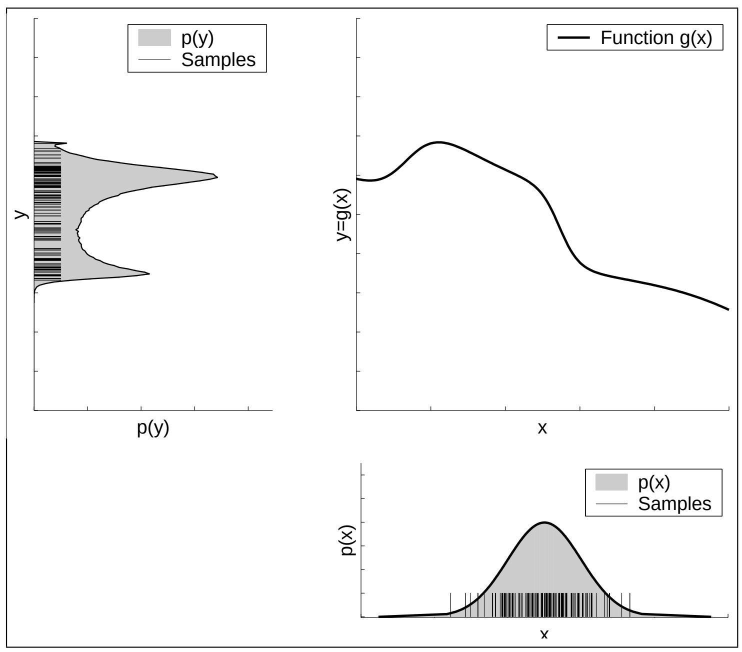


Figure 4.3 The "particle" representation used by particle filters. The lower right graph shows samples drawn from a Gaussian random variable, X. These samples are passed through the nonlinear function shown in the upper right graph. The resulting samples are distributed according to the random variable Y.

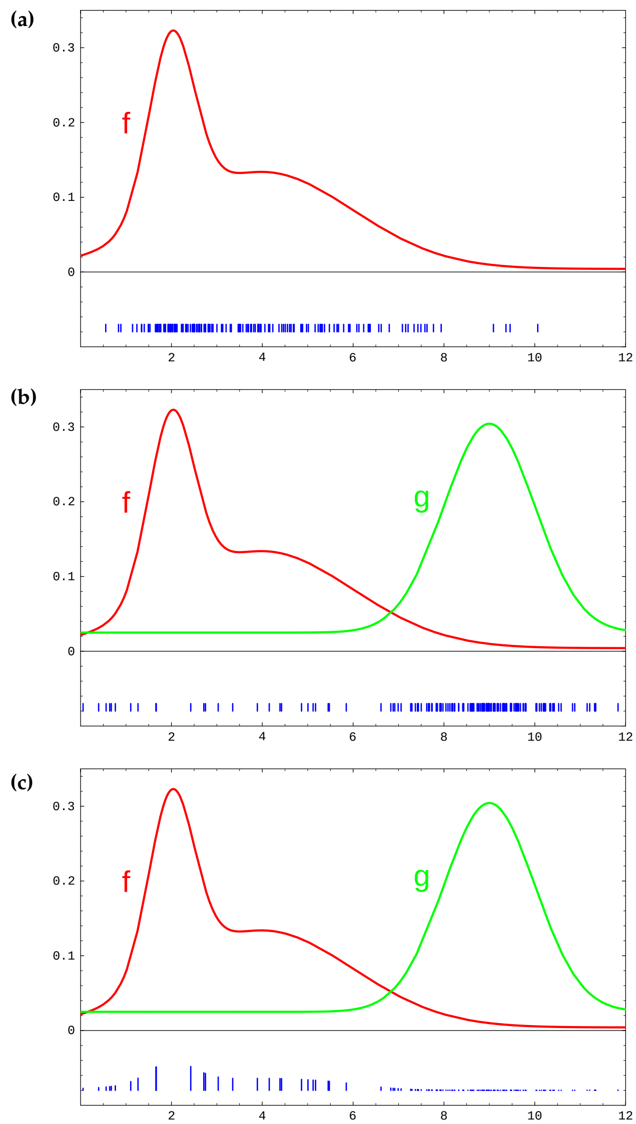


Figure 4.4 Illustration of importance factors in particle filters: (a) We seek to approximate the target density f. (b) Instead of sampling from f directly, we can only generate samples from a different density, g. Samples drawn from g are shown at the bottom of this diagram. (c) A sample of f is obtained by attaching the weight $f(x)/g(x)$ to each sample x. In particle filters, f corresponds to the belief $bel(x_t)$ and g to the belief $\overline{bel}(x_t)$.

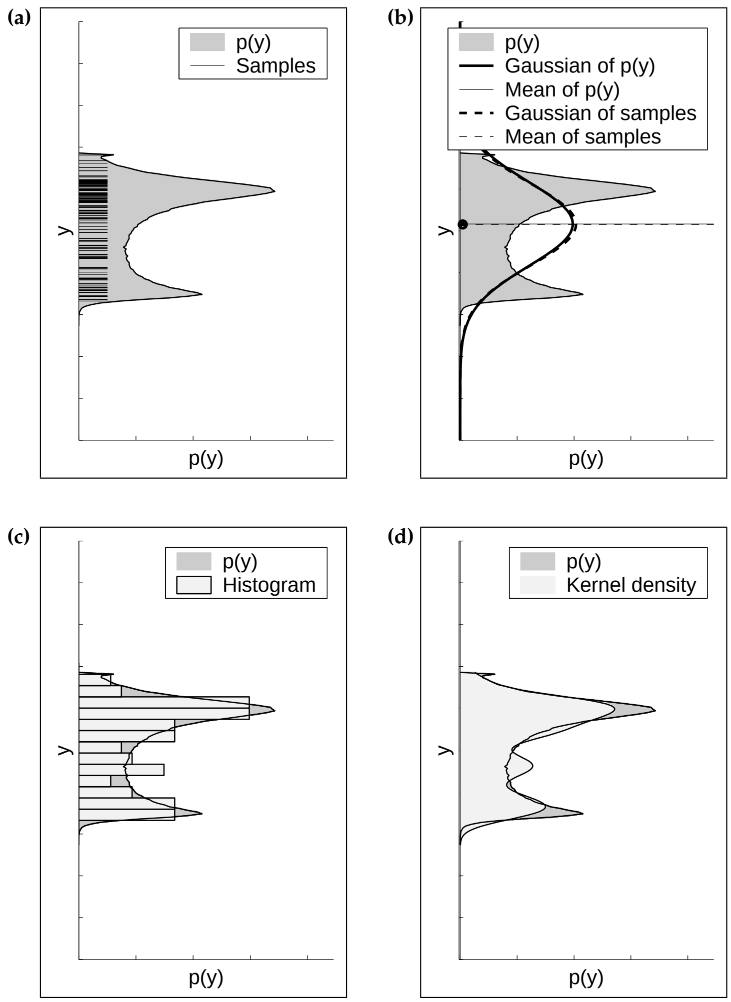


Figure 4.5 Different ways of extracting densities from particles. (a) Density and sample set approximation, (b) Gaussian approximation (mean and variance), (c) histogram approximation, (d) kernel density estimate. The choice of approximation strongly depends on the specific application and the computational resources.

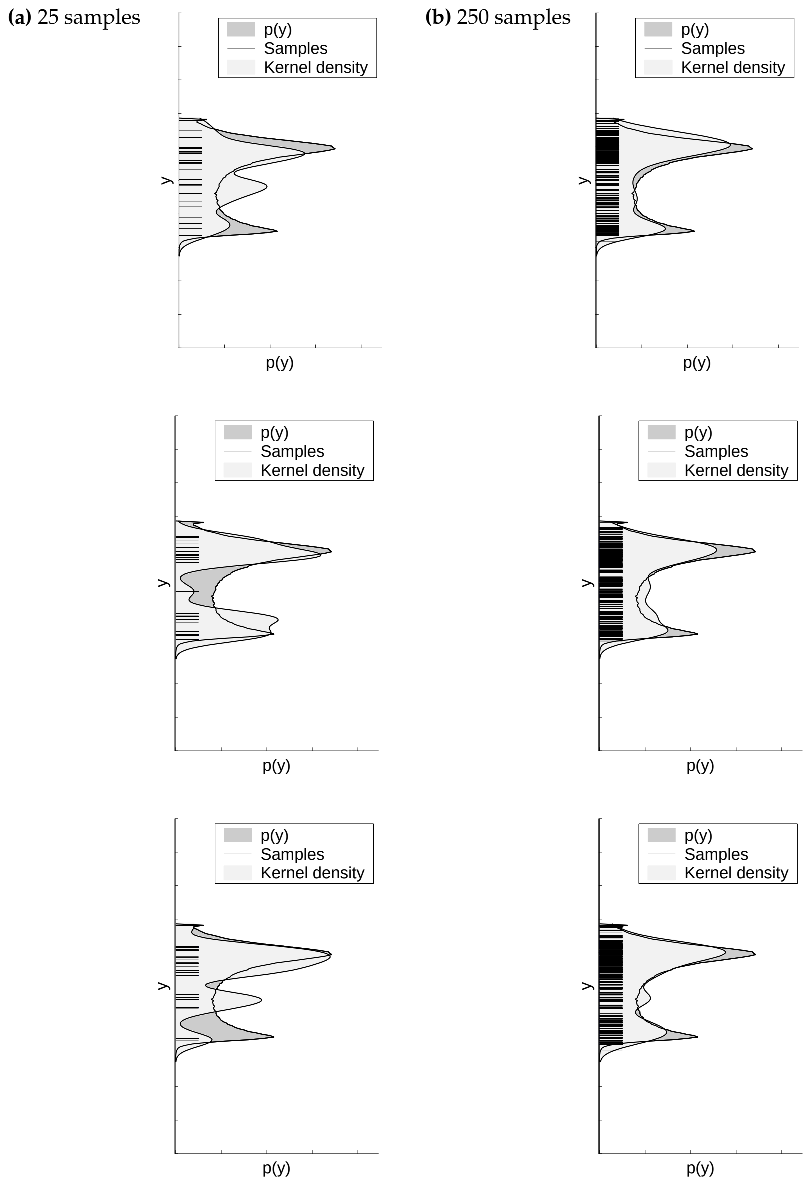


Figure 4.6 Variance due to random sampling. Samples are drawn from a Gaussian and passed through a nonlinear function. Samples and kernel estimates resulting from repeated sampling of 25 (left column) and 250 (right column) samples are shown. Each row shows one random experiment.

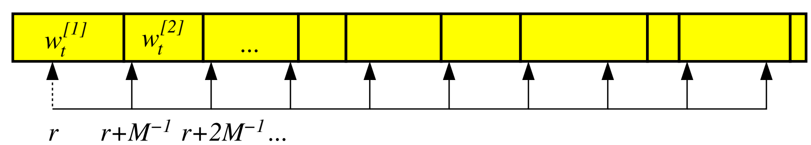


Figure 4.7 Principle of the low variance resampling procedure. We choose a random number r and then select those particles that correspond to $u = r + (m-1) \cdot M^{-1}$ where $m = 1, \ldots, M$.

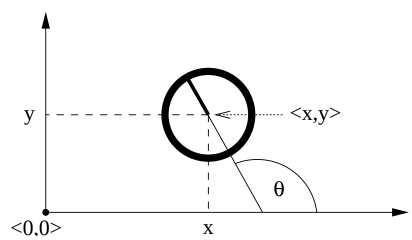


Figure 5.1 Robot pose, shown in a global coordinate system.

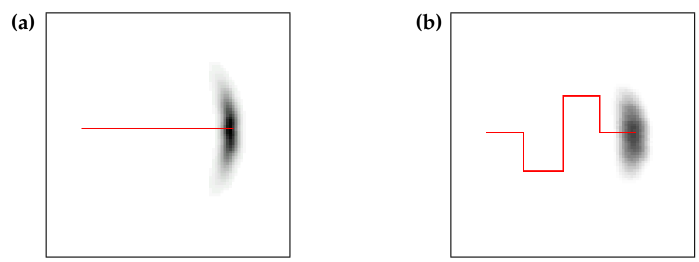


Figure 5.2 The motion model: Posterior distributions of the robot's pose upon executing the motion command illustrated by the solid line. The darker a location, the more likely it is. This plot has been projected into 2-D. The original density is three-dimensional, taking the robot's heading direction θ into account.

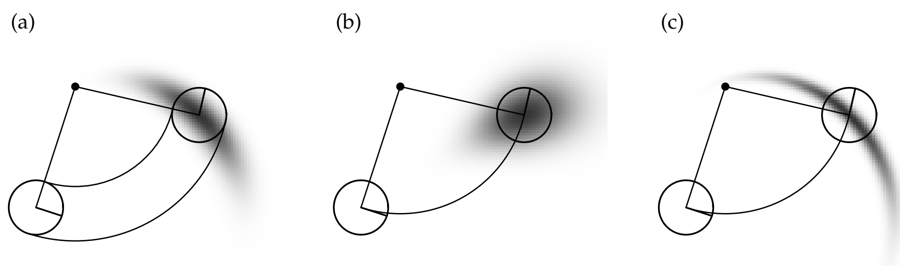


Figure 5.3 The velocity motion model, for different noise parameter settings.

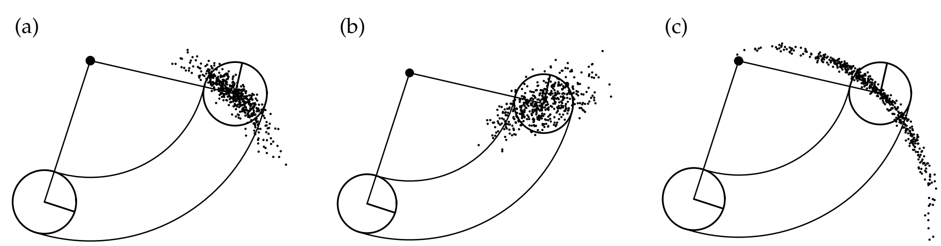


Figure 5.4 Sampling from the velocity motion model, using the same parameters as in Figure 5.3. Each diagram shows 500 samples.

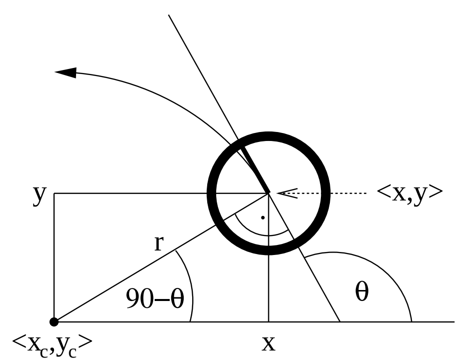


Figure 5.5 Motion carried out by a noise-free robot moving with constant velocities v and ω and starting at $(x \;\; y \;\; \theta)^T$.

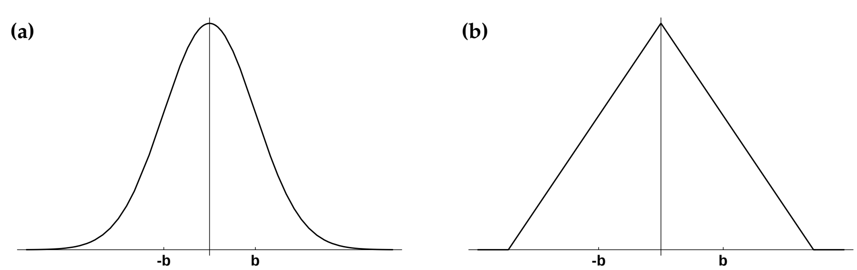


Figure 5.6 Probability density functions with standard deviation b: (a) Normal distribution, (b) triangular distribution.

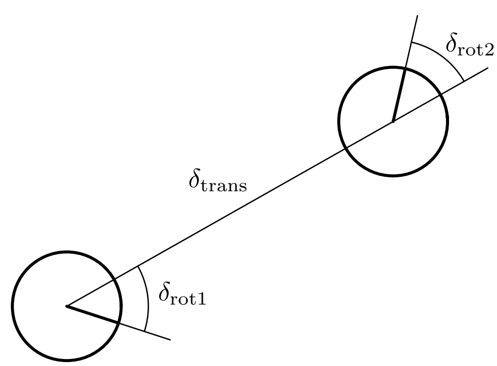


Figure 5.7 Odometry model: The robot motion in the time interval $(t-1, t]$ is approximated by a rotation δ_{rot1}, followed by a translation δ_{trans} and a second rotation δ_{rot2}. The turns and translations are noisy.

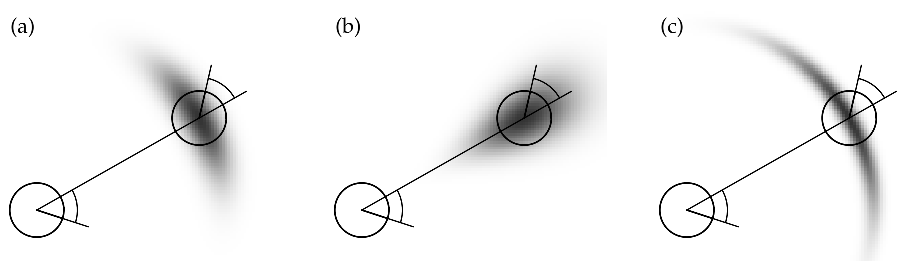


Figure 5.8 The odometry motion model, for different noise parameter settings.

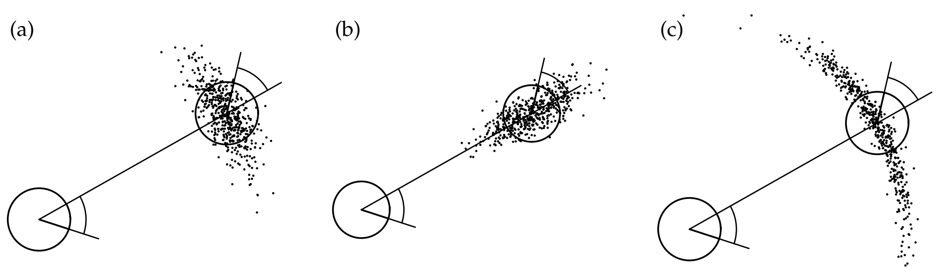


Figure 5.9 Sampling from the odometry motion model, using the same parameters as in Figure 5.8. Each diagram shows 500 samples.

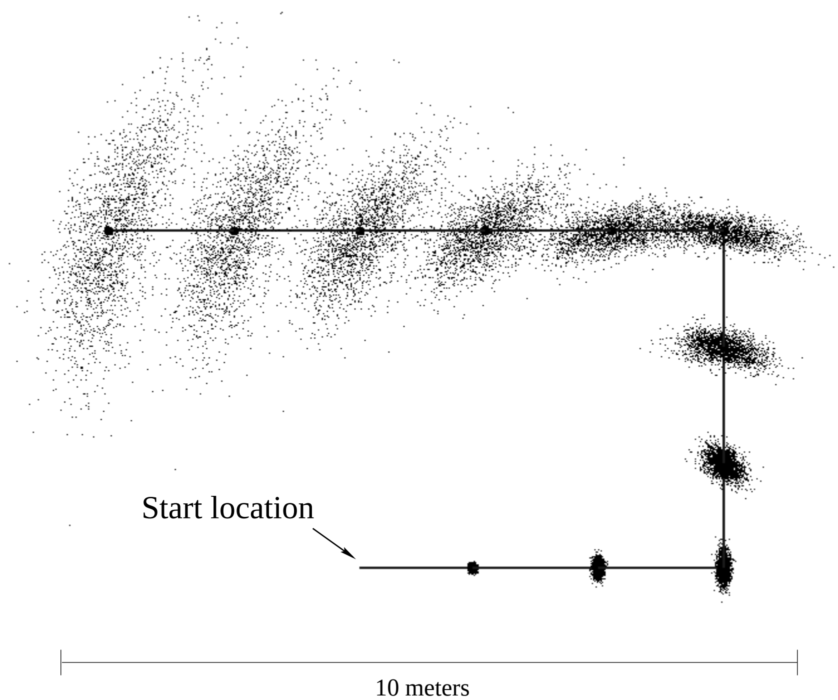


Figure 5.10 Sampling approximation of the position belief for a non-sensing robot. The solid line displays the actions, and the samples represent the robot's belief at different points in time.

(a) $p(x_t \mid u_t, x_{t-1})$ (b) $p(x_t \mid u_t, x_{t-1}, m)$

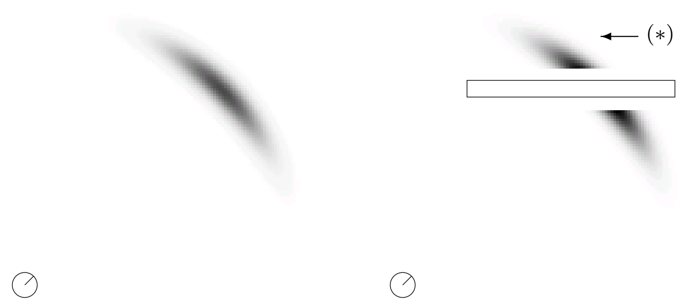


Figure 5.11 Velocity motion model (a) without a map and (b) conditioned on a map m.

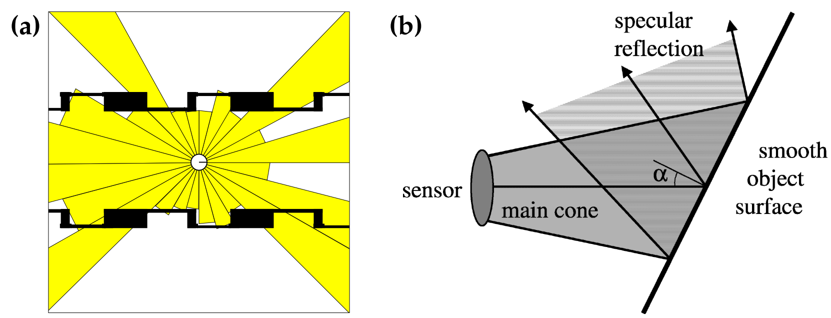


Figure 6.1 (a) Typical ultrasound scan of a robot in its environment. (b) A misreading in ultrasonic sensing. This effect occurs when firing a sonar signal towards a reflective surface at an angle α that exceeds half the opening angle of the sensor.

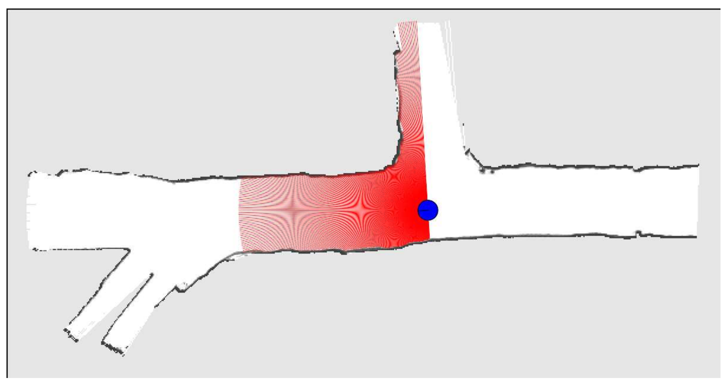


Figure 6.2 A typical laser range scan, acquired with a SICK LMS laser. The environment shown here is a coal mine. Image courtesy of Dirk Hähnel, University of Freiburg.

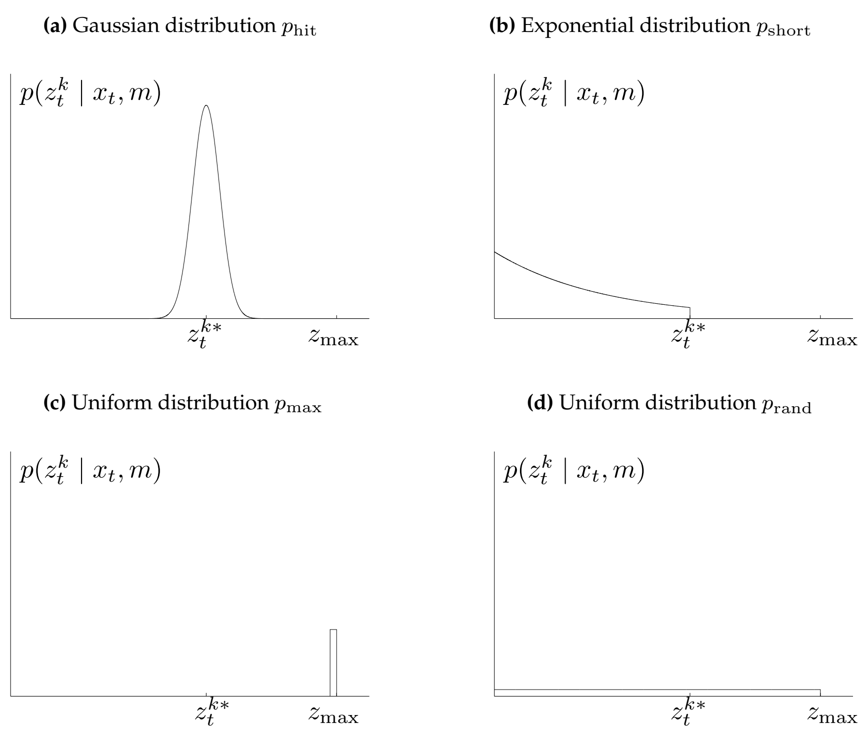


Figure 6.3 Components of the range finder sensor model. In each diagram the horizontal axis corresponds to the measurement z_t^k, the vertical to the likelihood.

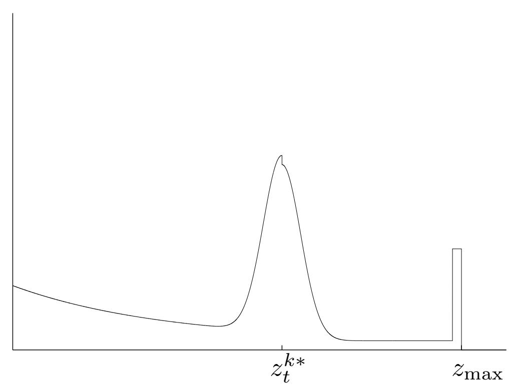


Figure 6.4 "Pseudo-density" of a typical mixture distribution $p(z_t^k \mid x_t, m)$.

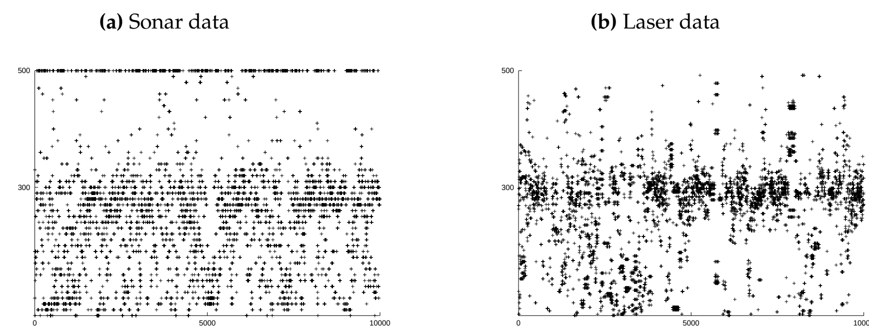


Figure 6.5 Typical data obtained with (a) a sonar sensor and (b) a laser-range sensor in an office environment for a "true" range of 300 cm and a maximum range of 500 cm.

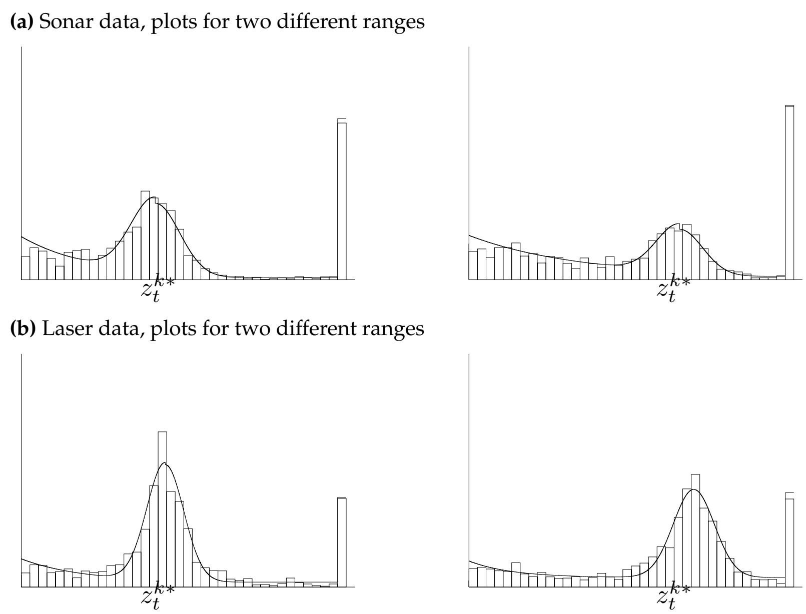


Figure 6.6 Approximation of the beam model based on (a) sonar data and (b) laser range data. The sensor models depicted on the left were obtained by a maximum likelihood approximation to the data sets depicted in Figure 6.5.

(a) Laser scan and part of the map

(b) Likelihood for different positions

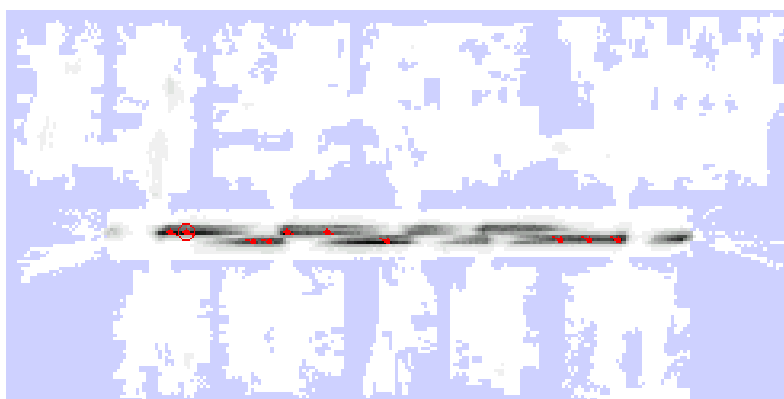


Figure 6.7 Probabilistic model of perception: (a) Laser range scan, projected into a previously acquired map m. (b) The likelihood $p(z_t \mid x_t, m)$, evaluated for all positions x_t and projected into the map (shown in gray). The darker a position, the larger $p(z_t \mid x_t, m)$.

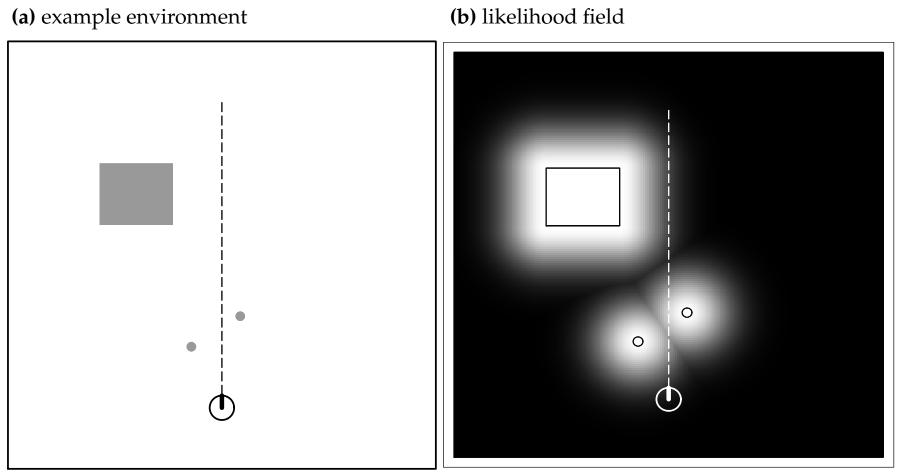


Figure 6.8 (a) Example environment with three obstacles (gray). The robot is located towards the bottom of the figure, and takes a measurement z_t^k as indicated by the dashed line. (b) Likelihood field for this obstacle configuration: the darker a location, the less likely it is to perceive an obstacle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

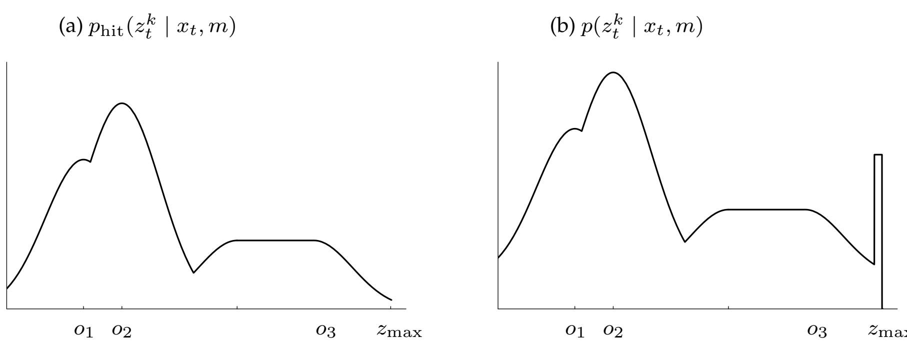


Figure 6.9 (a) Probability $p_{\text{hit}}(z_t^k)$ as a function of the measurement z_t^k, for the situation depicted in Figure 6.8. Here the sensor beam passes by three obstacles, with respective nearest points o_1, o_2, and o_3. (b) Sensor probability $p(z_t^k \mid x_t, m)$, obtained for the situation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

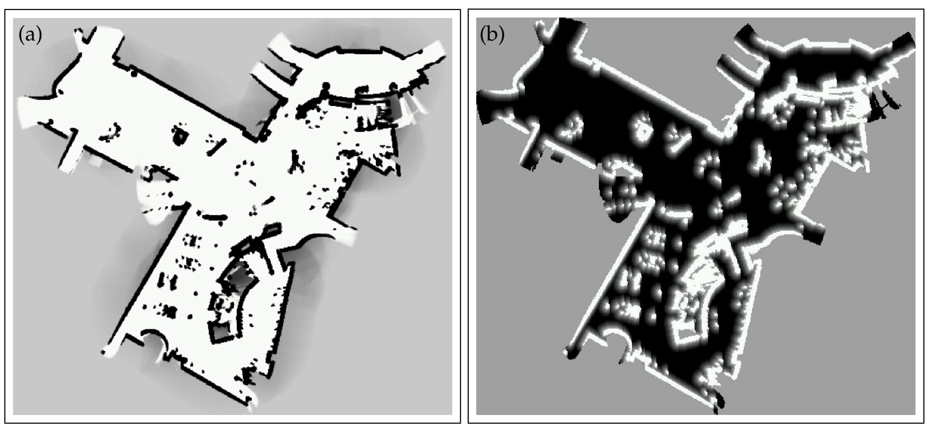


Figure 6.10 (a) Occupancy grid map of the San Jose Tech Museum, (b) pre-processed likelihood field.

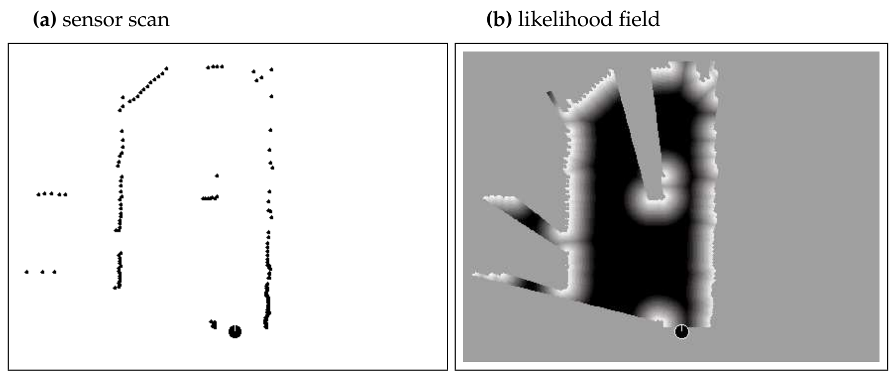


Figure 6.11 (a) Sensor scan, from a bird's eye perspective. The robot is placed at the bottom of this figure, generating a proximity scan that consists of the 180 dots in front of the robot. (b) Likelihood function generated from this sensor scan. The darker a region, the smaller the likelihood for sensing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

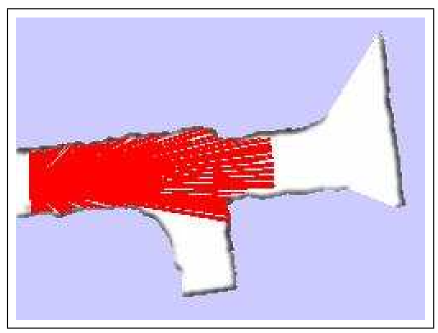


Figure 6.12 Example of a local map generated from 10 range scans, one of which is shown.

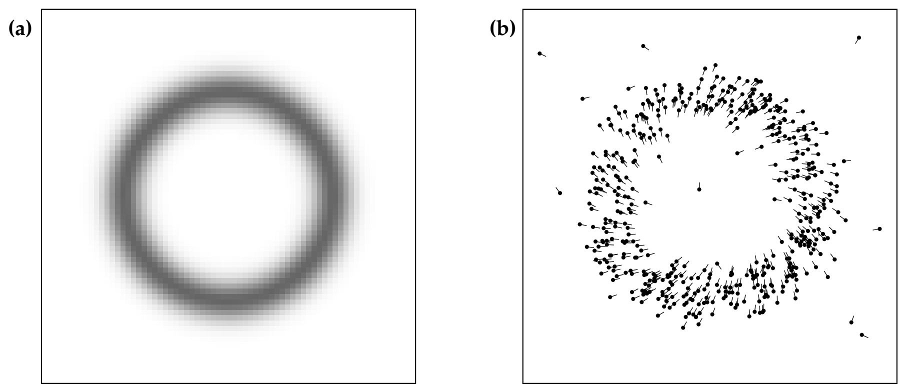


Figure 6.13 Landmark detection model: (a) Posterior distribution of the robot's pose given that it detected a landmark in 5m distance and 30deg relative bearing (projected onto 2-D). (b) Sample robot poses generated from such a detection. The lines indicate the orientation of the poses.

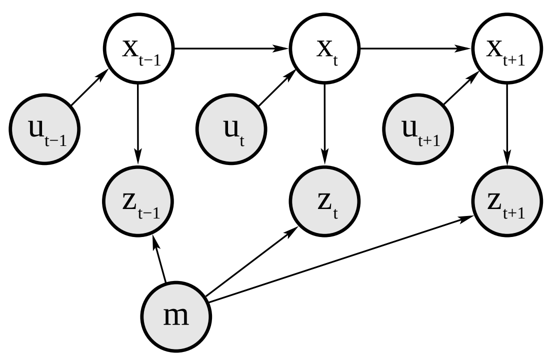


Figure 7.1 Graphical model of mobile robot localization. The value of shaded nodes are known: the map m, the measurements z, and the controls u. The goal of localization is to infer the robot pose variables x.

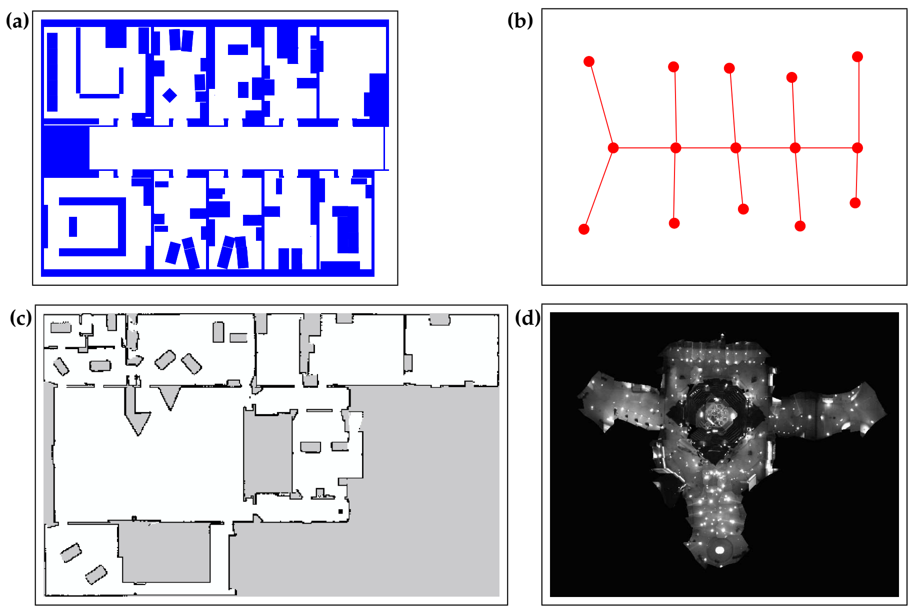


Figure 7.2 Example maps used for robot localization: (a) a manually constructed 2-D metric layout, (b) a graph-like topological map, (c) an occupancy grid map, and (d) an image mosaic of a ceiling. (d) courtesy of Frank Dellaert, Georgia Institute of Technology.

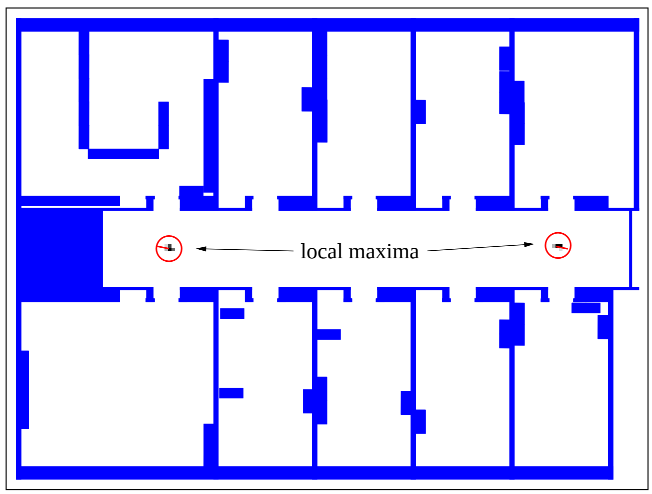


Figure 7.3 Example situation that shows a typical belief state during global localization in a locally symmetric environment. The robot has to move into one of the rooms to determine its location.

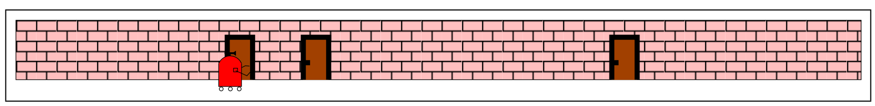


Figure 7.4 Example environment used to illustrate mobile robot localization: One-dimensional hallway environment with three indistinguishable doors. Initially the robot does not know its location except for its heading direction. Its goal is to find out where it is.

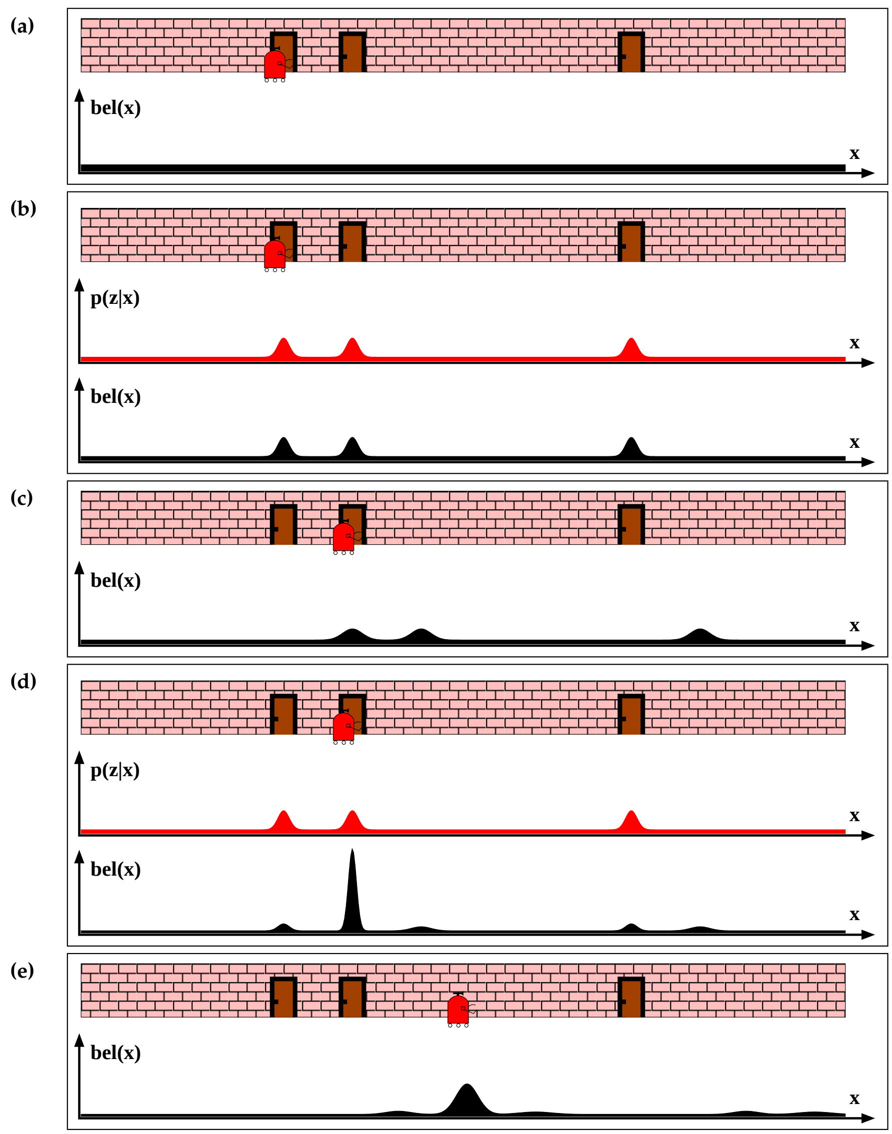


Figure 7.5 Illustration of the Markov localization algorithm. Each picture depicts the position of the robot in the hallway and its current belief $bel(x)$. (b) and (d) additionally depict the observation model $p(z_t \mid x_t)$, which describes the probability of observing a door at the different locations in the hallway.

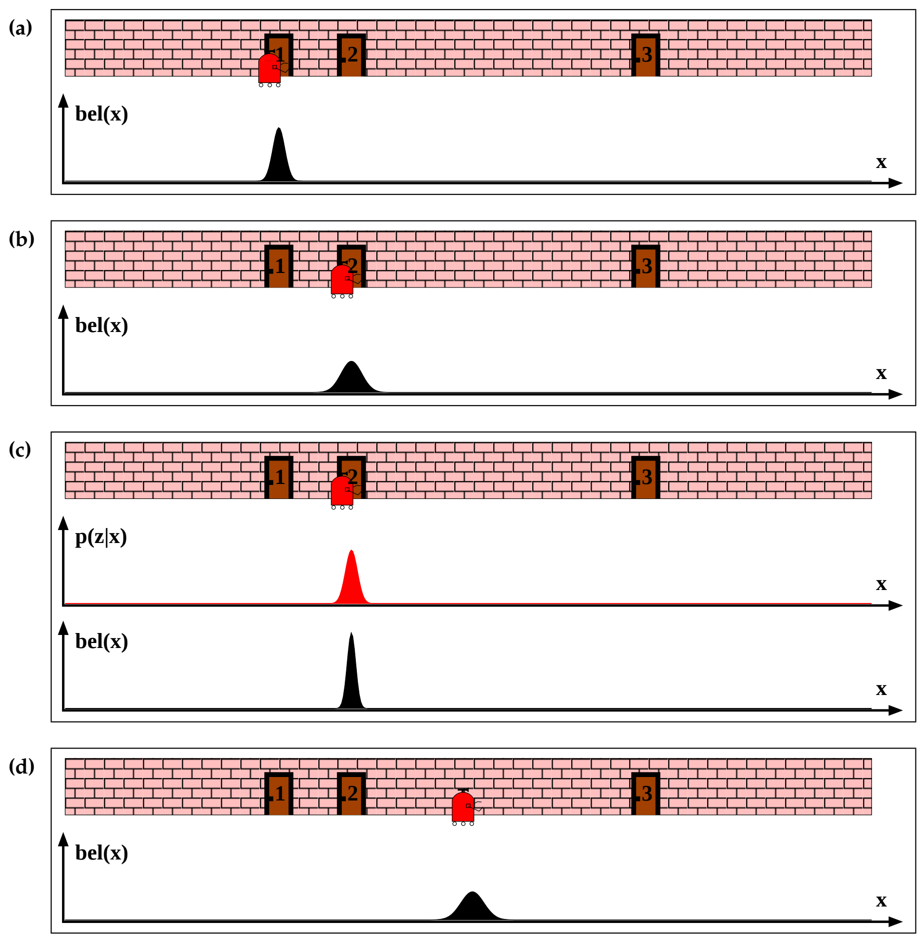


Figure 7.6 Application of the Kalman filter algorithm to mobile robot localization. All densities are represented by unimodal Gaussians.

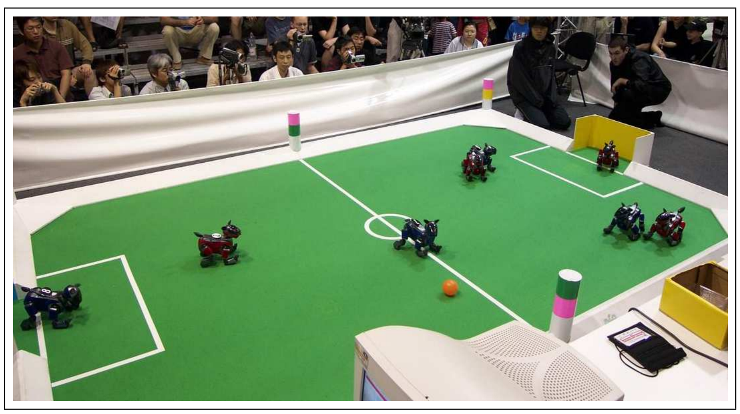


Figure 7.7 AIBO robots on the RoboCup soccer field. Six landmarks are placed at the corners and the midlines of the field.

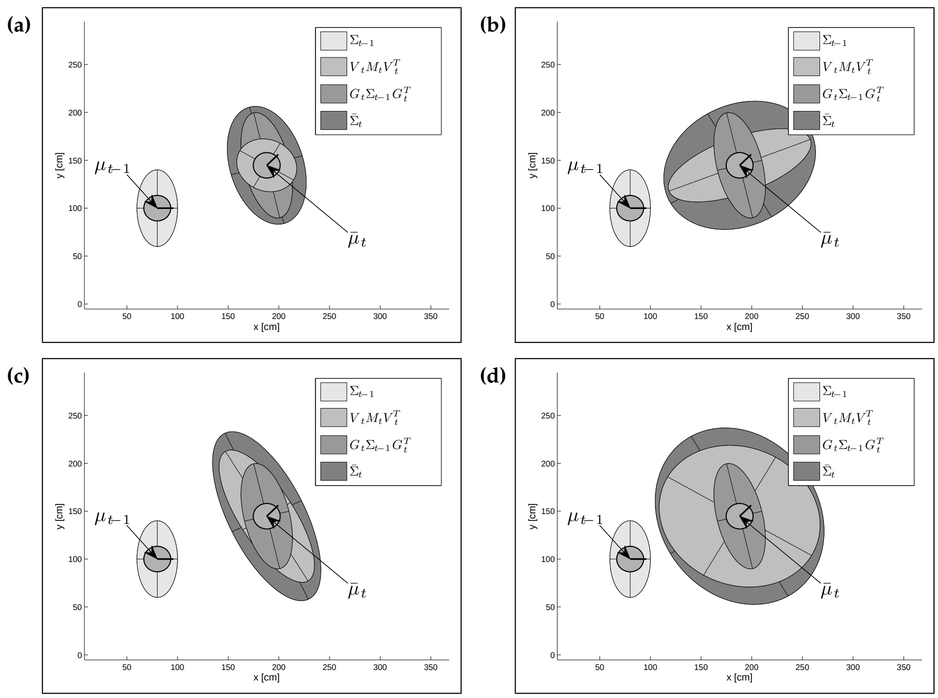


Figure 7.8 Prediction step of the EKF algorithm. The panels were generated with different motion noise parameters. The robot's initial estimate is represented by the ellipse centered at μ_{t-1}. After moving on a circular arc of 90cm length while turning 45 degrees to the left, the predicted position is centered at $\bar{\mu}_t$. In panel (a), the motion noise is relatively small in both translation and rotation. The other panels represent (b) high translational noise, (c) high rotational noise, and (d) high noise in both translation and rotation.

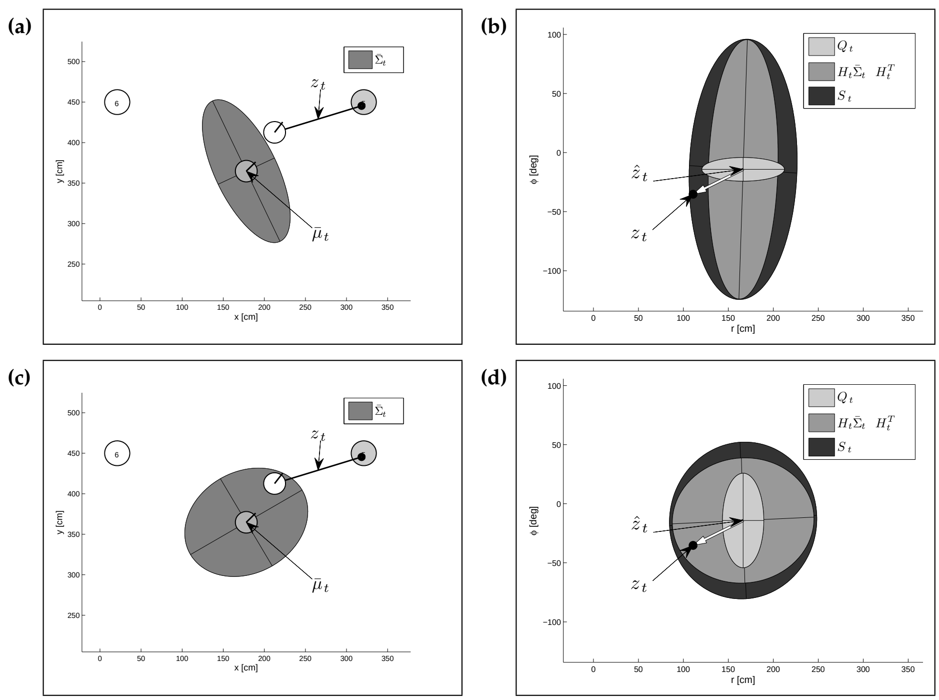


Figure 7.9 Measurement prediction. The left plots show two predicted robot locations along with their uncertainty ellipses. The true robot and the observation are indicated by the white circle and the bold line, respectively. The panels on the right show the resulting measurement predictions. The white arrows indicate the innovations, the differences between observed and predicted measurements.

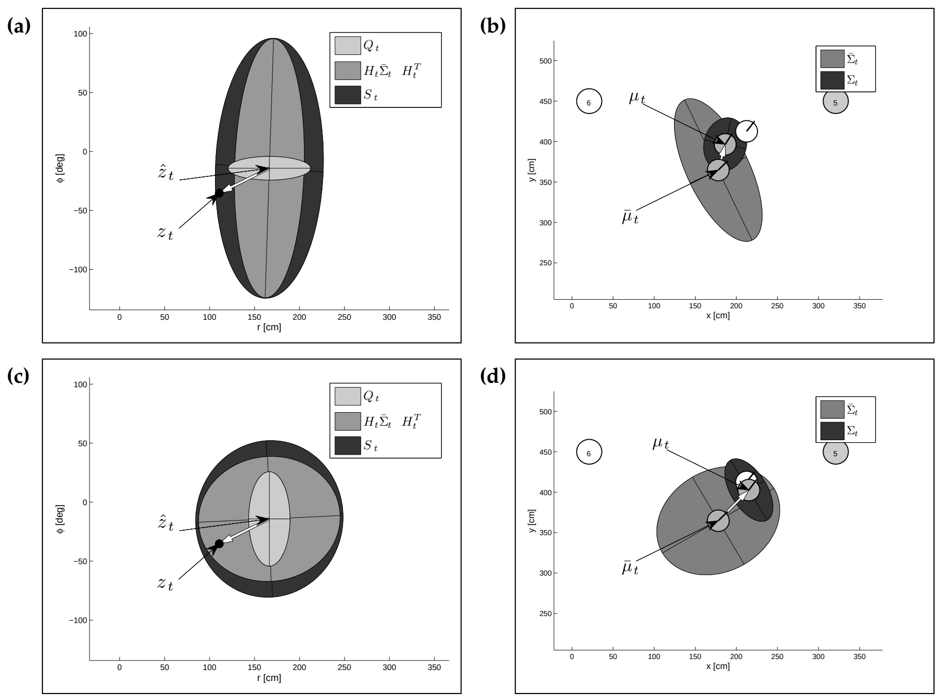


Figure 7.10 Correction step of the EKF algorithm. The panels on the left show the measurement prediction, and the panels on the right the resulting corrections, which update the mean estimate and reduce the position uncertainty ellipses.

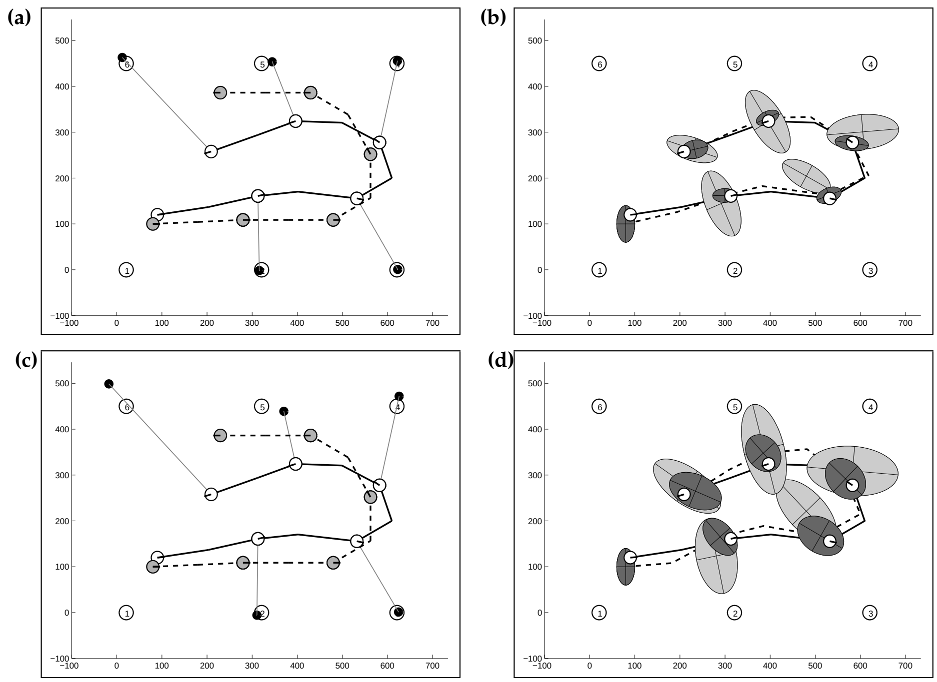


Figure 7.11 EKF-based localization with an accurate (upper row) and a less accurate (lower row) landmark detection sensor. The dashed lines in the left panel indicate the robot trajectories as estimated from the motion controls. The solid lines represent the true robot motion resulting from these controls. Landmark detections at five locations are indicated by the thin lines. The dashed lines in the right panels show the corrected robot trajectories, along with uncertainty before (light gray, $\bar{\Sigma}_t$) and after (dark gray, Σ_t) incorporating a landmark detection.

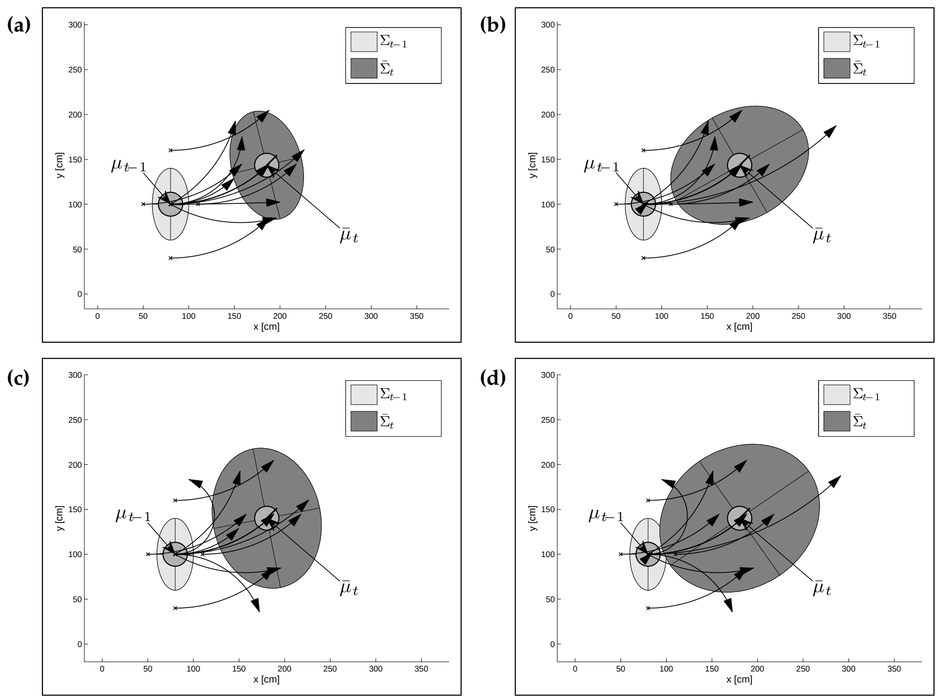


Figure 7.12 Prediction step of the UKF algorithm. The graphs were generated with different motion noise parameters. The robot's initial estimate is represented by the ellipse centered at μ_{t-1}. The robot moves on a circular arc of 90cm length while turning 45 degrees to the left. In panel (a), the motion noise is relatively small in both translation and rotation. The other panels represent (b) high translational noise, (c) high rotational noise, and (d) high noise in both translation and rotation.

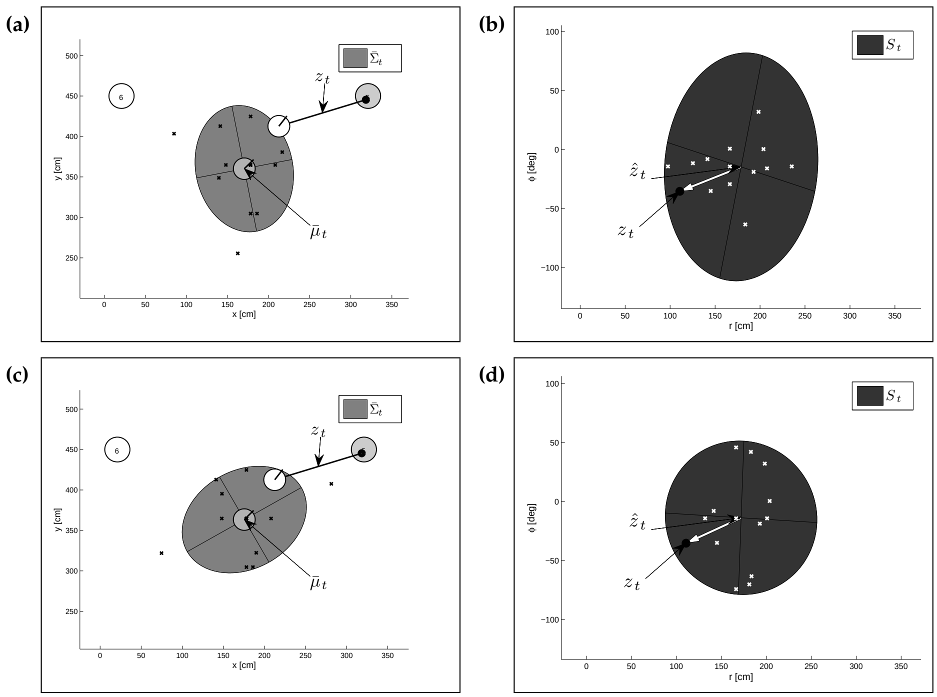


Figure 7.13 Measurement prediction. The left plots show the sigma points predicted from two motion updates along with the resulting uncertainty ellipses. The true robot and the observation are indicated by the white circle and the bold line, respectively. The panels on the right show the resulting measurement prediction sigma points. The white arrows indicate the innovations, the differences between observed and predicted measurements.

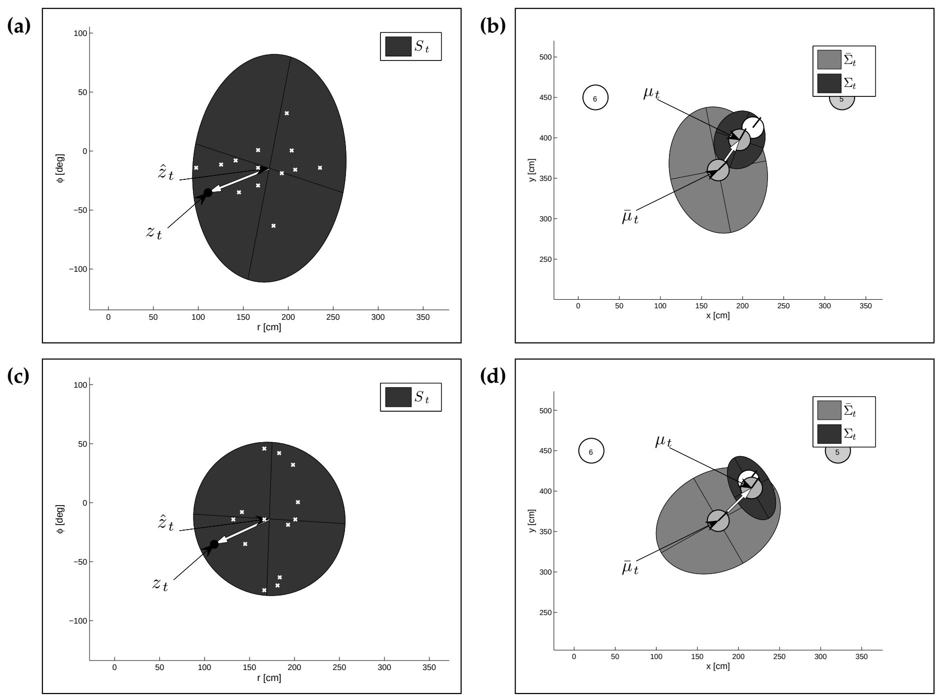


Figure 7.14 Correction step of the UKF algorithm. The panels on the left show the measurement prediction, and the panels on the right the resulting corrections, which update the mean estimate and reduce the position uncertainty ellipses.

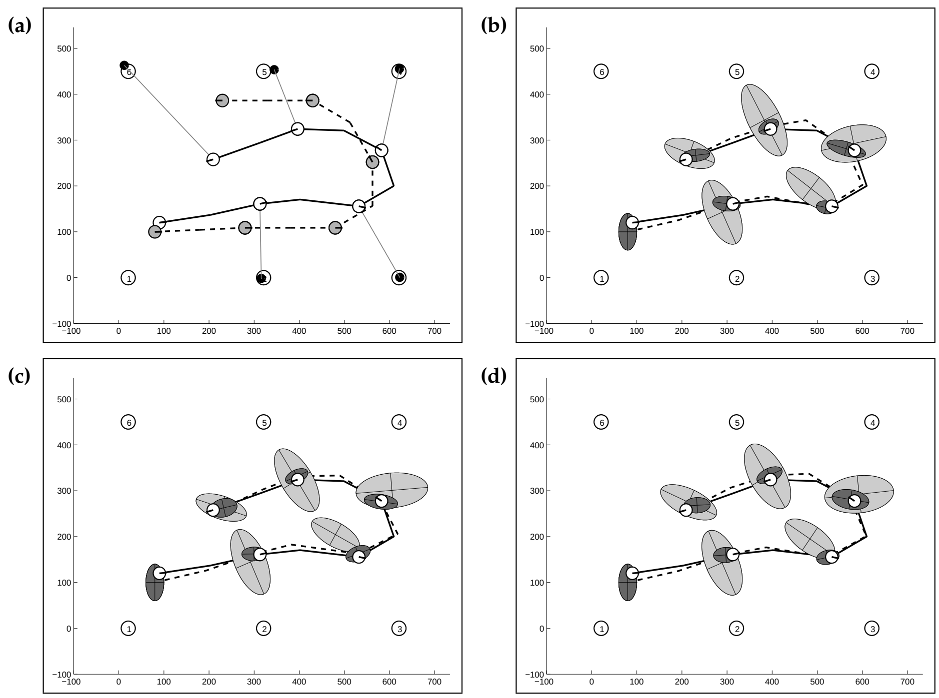


Figure 7.15 Comparison of UKF and EKF estimates: (a) Robot trajectory according to the motion control (dashed lines) and the resulting true trajectory (solid lines). Landmark detections are indicated by thin lines. (b) Reference estimates, generated by a particle filter. (c) EKF and (d) UKF estimates.

(a)

(b)

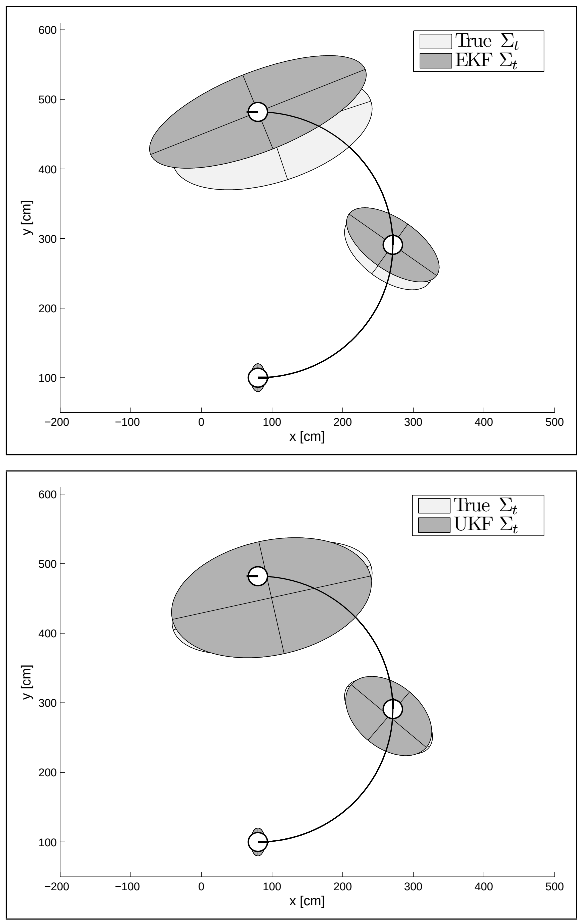


Figure 7.16 Approximation error due to linearization. The robot moves on a circle. Estimates based on (a) EKF prediction and (b) UKF prediction. The reference covariances are extracted from an accurate, sample-based prediction.

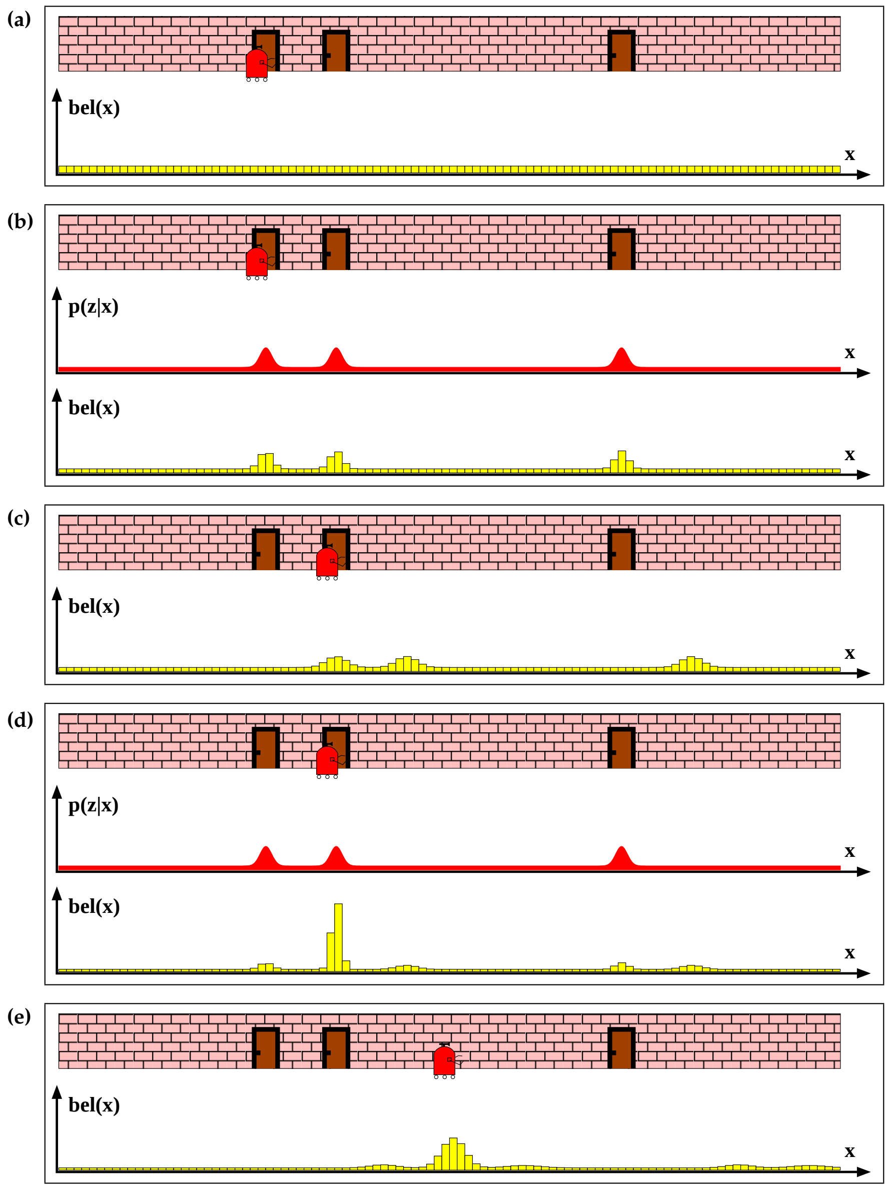


Figure 8.1 Grid localization using a fine-grained metric decomposition. Each picture depicts the position of the robot in the hallway along with its belief $bel(x_t)$, represented by a histogram over a grid.

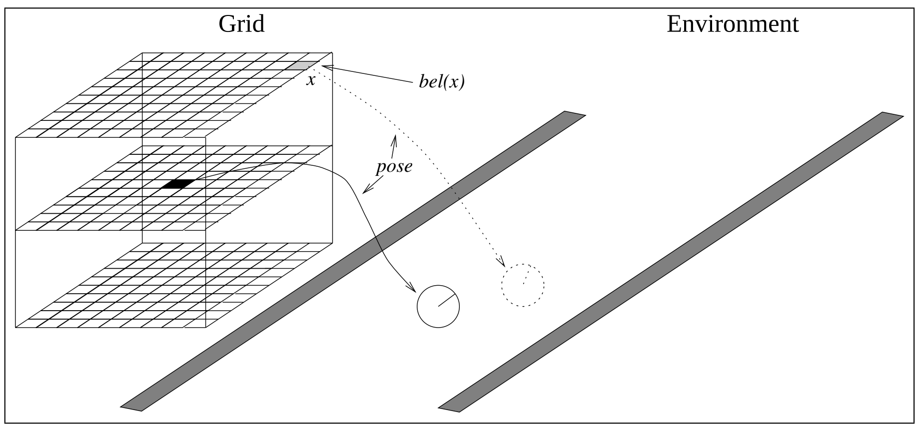


Figure 8.2 Example of a fixed-resolution grid over the robot pose variables x, y, and θ. Each grid cell represents a robot pose in the environment. Different orientations of the robot correspond to different planes in the grid (shown are only three orientations).

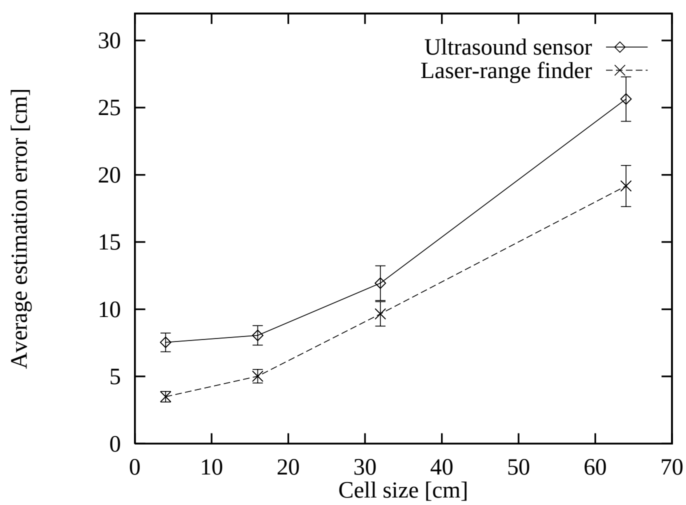


Figure 8.3 Average localization error as a function of grid cell size, for ultrasound sensors and laser range-finders.

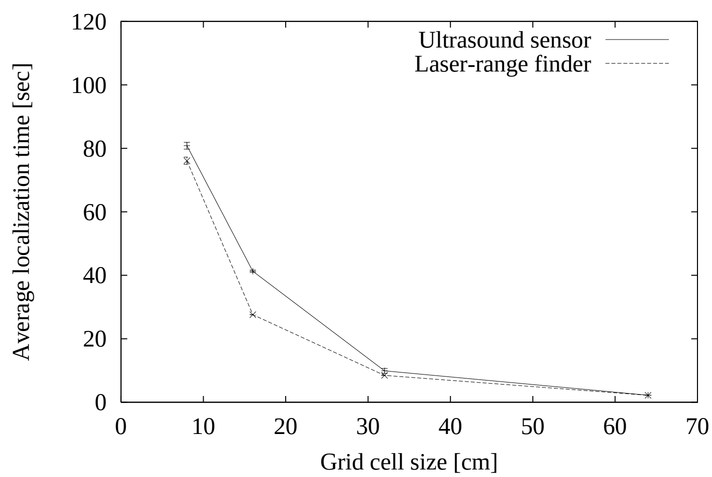


Figure 8.4 Average CPU-time needed for global localization as a function of grid resolution, shown for both ultrasound sensors and laser range-finders.

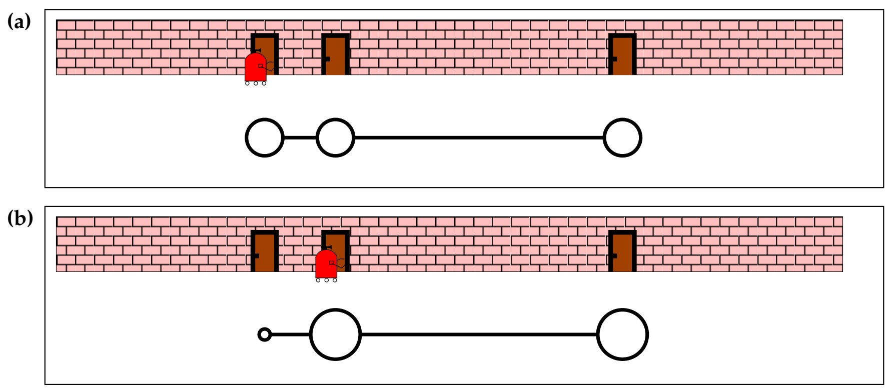


Figure 8.5 Application of a coarse-grained, topological representation to mobile robot localization. Each state corresponds to a distinctive place in the environment (a door in this case). The robot's belief $bel(x_t)$ of being in a state is represented by the size of the circles. (a) The initial belief is uniform over all poses. (b) shows the belief after the robot made one state transition and detected a door. At this point, it is unlikely that the robot is still in the left position.

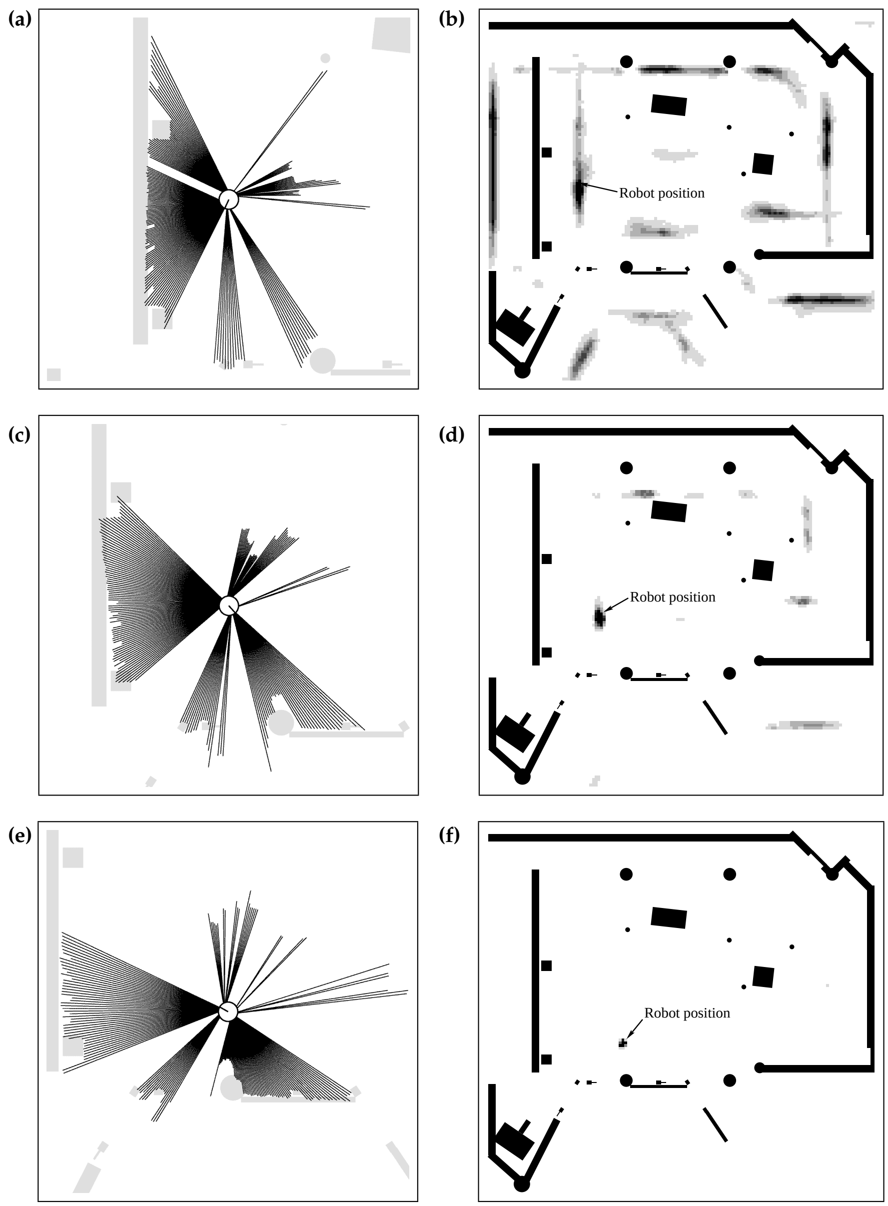


Figure 8.6 Global localization in a map using laser range-finder data. (a) Scan of the laser range-finders taken at the start position of the robot (max range readings are omitted). Figure (b) shows the situation after incorporating this laser scan, starting with the uniform distribution. (c) Second scan and (d) resulting belief. After integrating the final scan shown in (e), the robot's belief is centered at its actual location (see (f)).

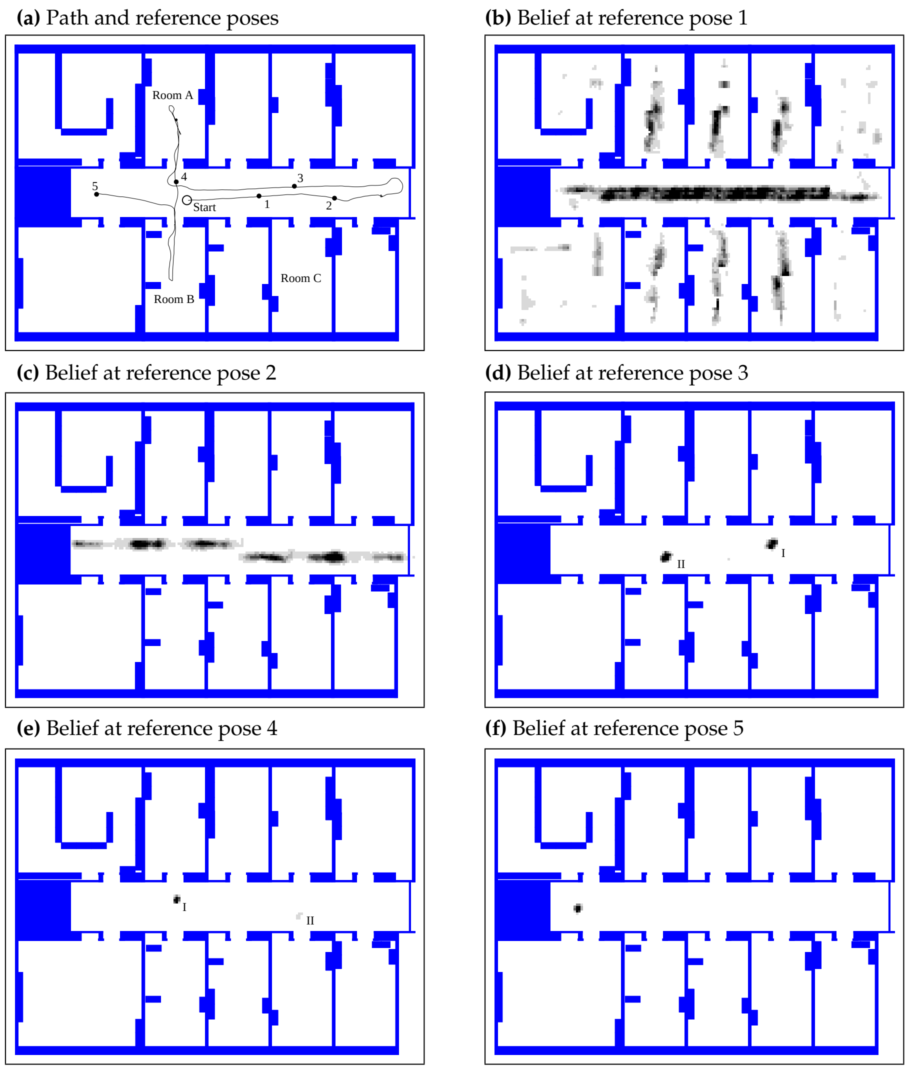


Figure 8.7 Global localization in an office environment using sonar data. (a) Path of the robot. (b) Belief as the robot passes position `1`. (c) After some meters of robot motion, the robot knows that it is in the corridor. (d) As the robot reaches position `3` it has scanned the end of the corridor with its sonar sensors and hence the distribution is concentrated on two local maxima. While the maximum labeled `I` represents the true location of the robot, the second maximum arises due to the symmetry of the corridor (position `II` is rotated by 180° relative to position `I`). (e) After moving through `Room A`, the probability of being at the correct position `I` is now higher than the probability of being at position `II`. (f) Finally the robot's belief is centered on the correct pose.

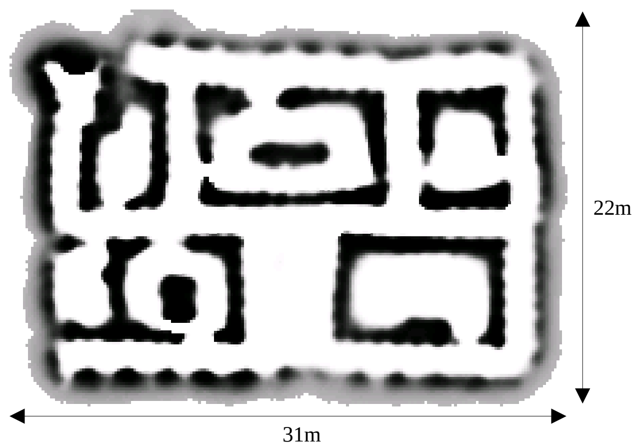


Figure 8.8 Occupancy grid map of the 1994 AAAI mobile robot competition arena.

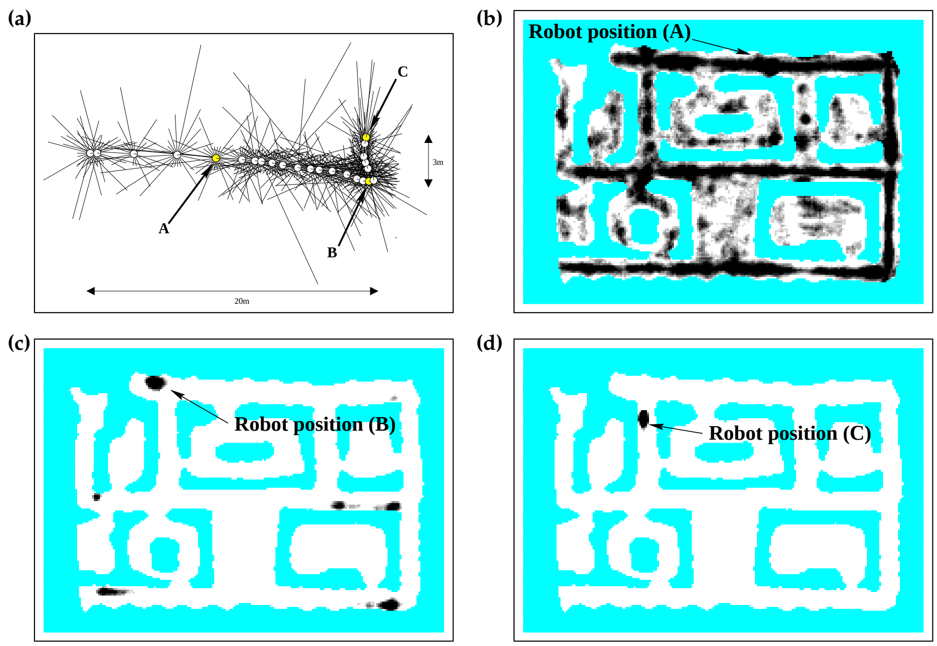


Figure 8.9 (a) Data set (odometry and sonar range scans) collected in the environment shown in Figure 8.8. This data set is sufficient for global localization using the grid localization. The beliefs at the points marked "A," "B" and "C" are shown in (b), (c), and (d).

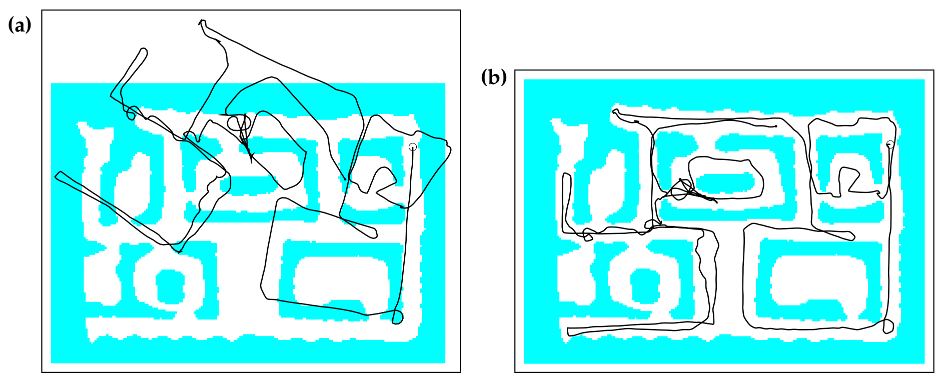


Figure 8.10 (a) Odometry information and (b) corrected path of the robot.

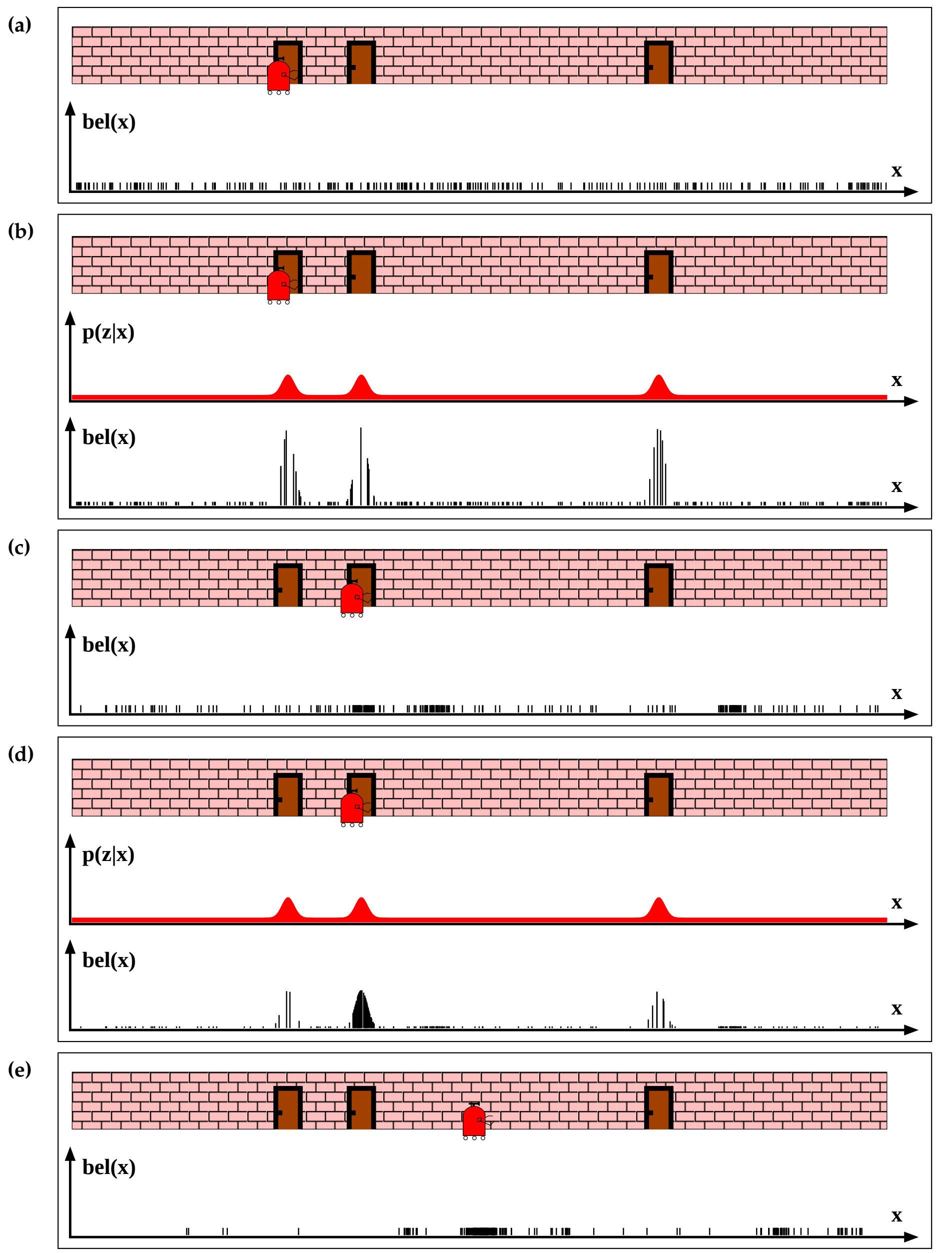


Figure 8.11 Monte Carlo Localization, a particle filter applied to mobile robot localization.

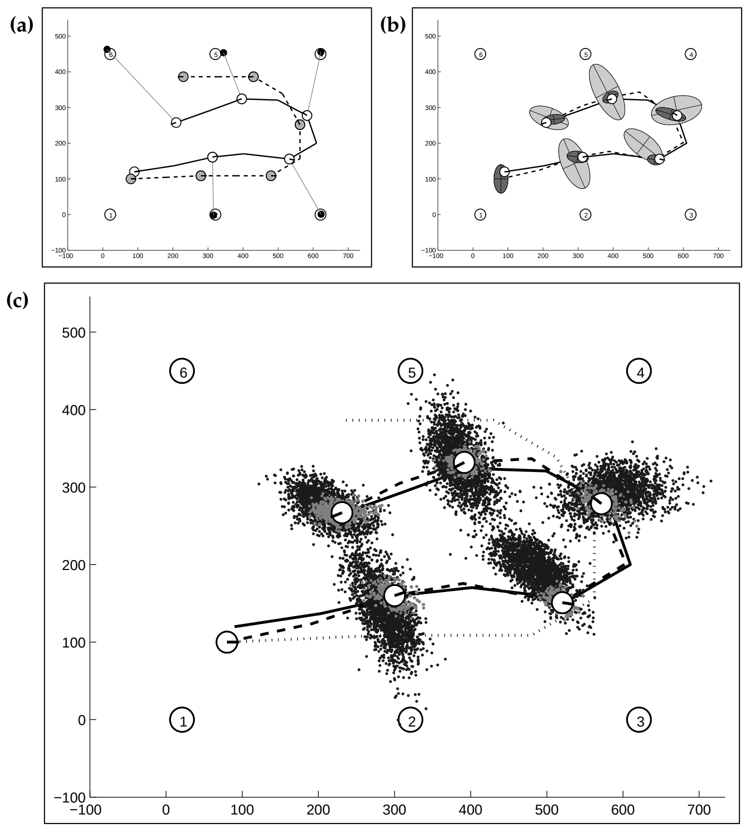


Figure 8.12 The MCL algorithm for landmark-based localization. (a) Robot trajectory according to the motion control (dashed lines) and the resulting true trajectory (solid lines). Landmark detections are indicated by thin lines. (b) Covariances of sample sets before and after resampling. (c) Sample sets before and after resampling.

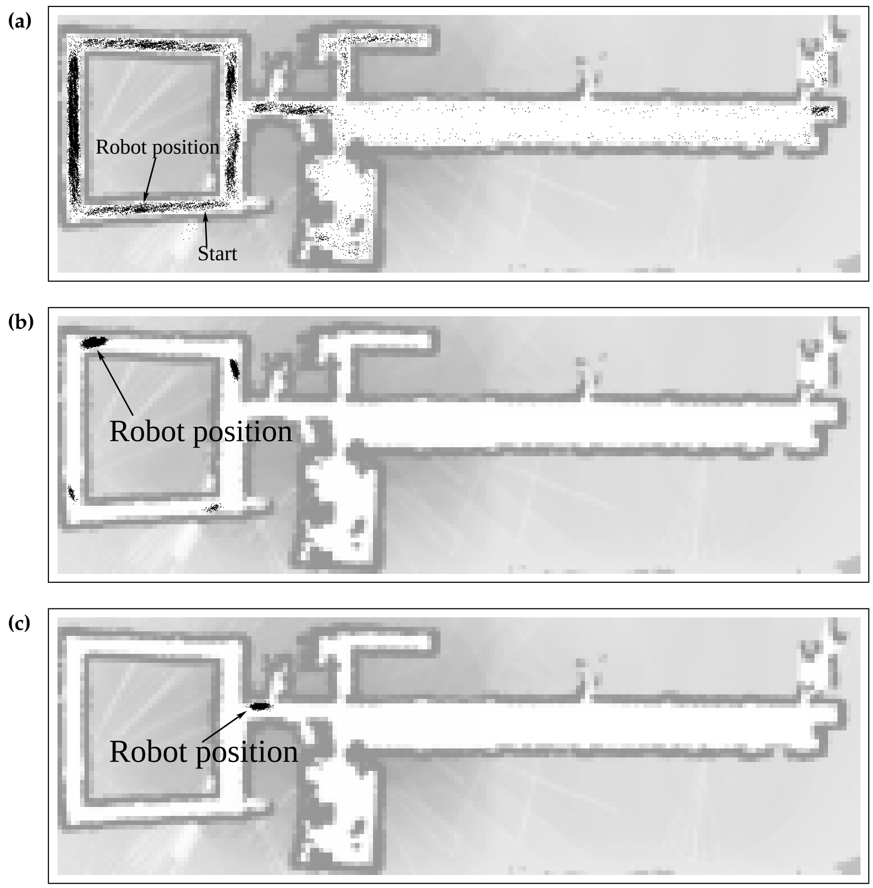


Figure 8.13 Illustration of Monte Carlo localization: Shown here is a robot operating in an office environment of size 54m $\times$ 18m. (a) After moving 5m, the robot is still globally uncertain about its position and the particles are spread through major parts of the free-space. (b) Even as the robot reaches the upper left corner of the map, its belief is still concentrated around four possible locations. (c) Finally, after moving approximately 55m, the ambiguity is resolved and the robot knows where it is. All computation is carried out in real-time on a low-end PC.

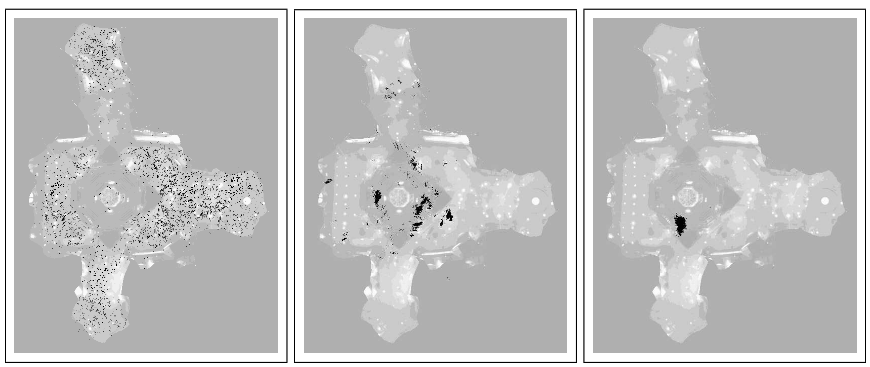


Figure 8.14 Global localization using a camera pointed at the ceiling.

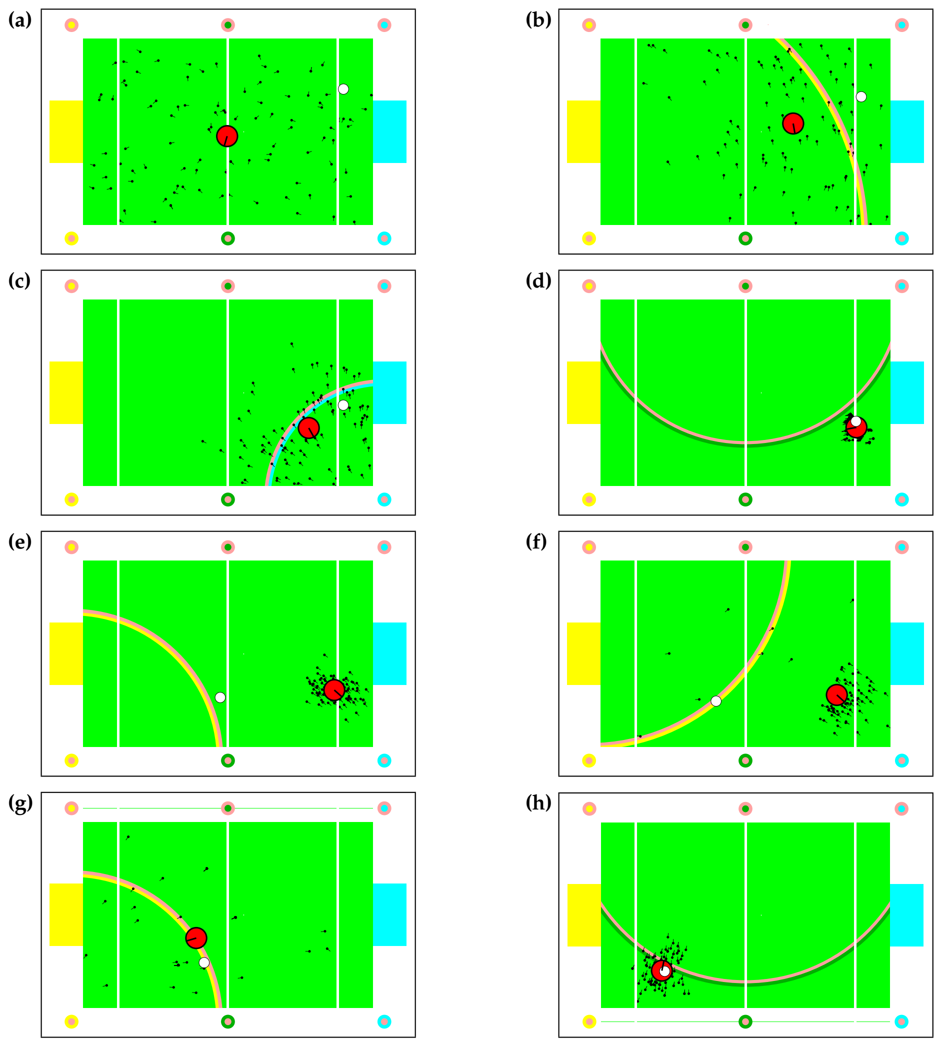


Figure 8.16 Monte Carlo localization with random particles. Each picture shows a particle set representing the robot's position estimate (small lines indicate the orientation of the particles). The large circle depicts the mean of the particles, and the true robot position is indicated by the small, white circle. Marker detections are illustrated by arcs centered at the detected marker. The pictures illustrate global localization (a)–(d) and relocalization (e)–(h).

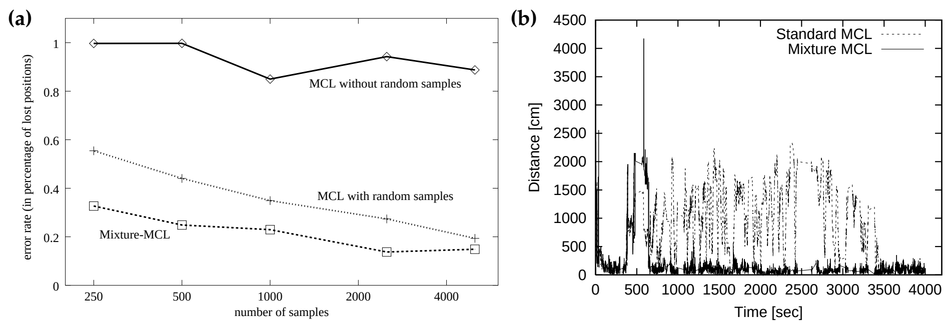


Figure 8.17 (a) plain MCL (top curve), MCL with random samples (center curve), and *Mixture MCL* with mixture proposal distribution (bottom curve). The error rate is measured in percentage of time during which the robot lost track of its position, for a data set acquired by a robot operating in a crowded museum. (b) Error as a function of time for standard MCL and mixture MCL, using a ceiling map for localization.

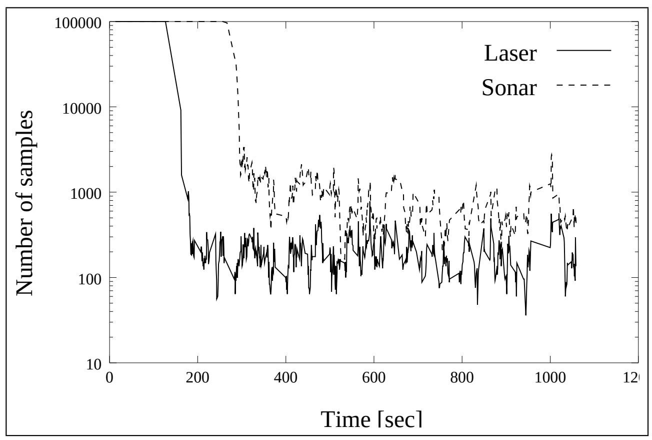


Figure 8.18 KLD-sampling: Typical evolution of number of samples for a global localization run, plotted against time (number of samples is shown on a log scale). The solid line shows the number of samples when using the robot's laser range-finder, the dashed graph is based on sonar sensor data.

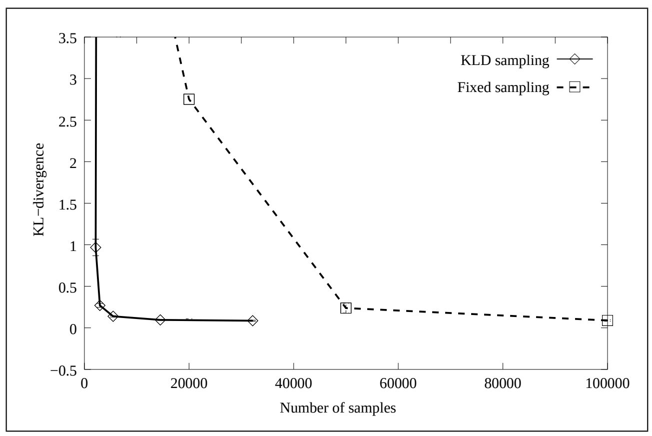


Figure 8.19 Comparison of KLD-sampling and MCL with fixed sample set sizes. The x-axis represents the average sample set size. The y-axis plots the KL-distance between the reference beliefs and the sample sets generated by the two approaches.

Figure 8.20 Scenes from the *"Deutsches Museum Bonn,"* where the mobile robot "Rhino" was frequently surrounded by dozens of people.

(a) (b)

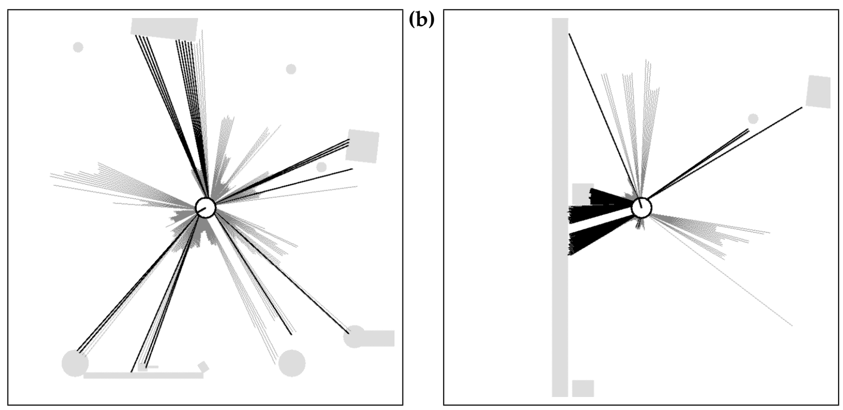


Figure 8.21 Laser range scans are often heavily corrupted when people surround the robot. How can a robot maintain accurate localization under such circumstances?

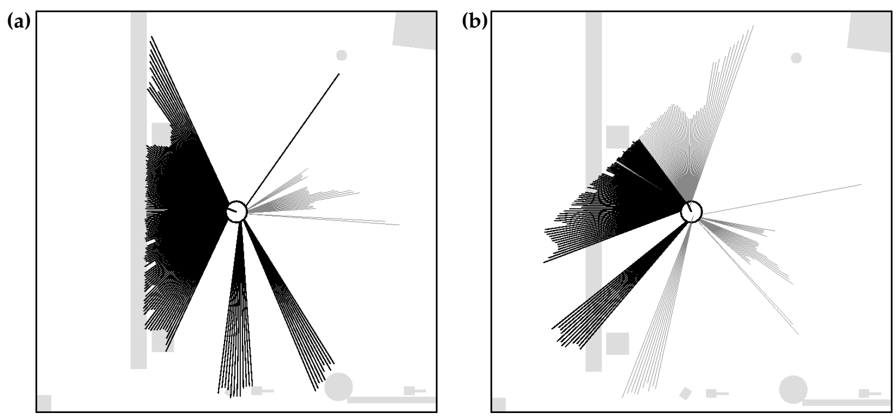


Figure 8.22 Illustration of our measurement rejection algorithm: Shown in both diagrams are range scans (no max-range readings). Lightly shaded readings are filtered out.

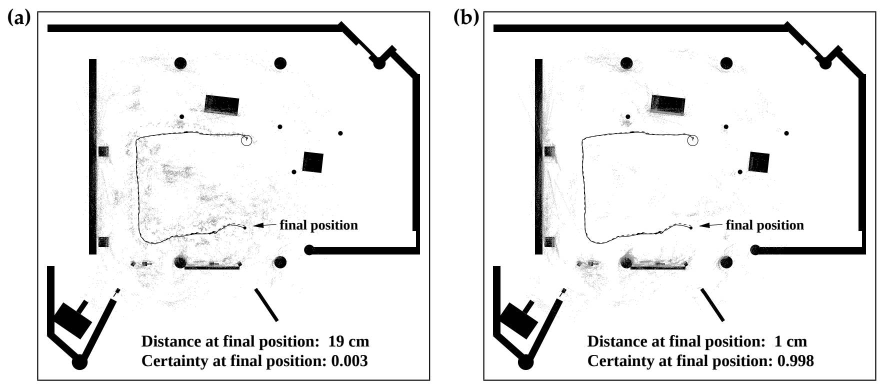


Figure 8.23 Comparison of (a) standard MCL and (b) MCL with the removal of sensor measurements likely caused by unexpected obstacles. Both diagrams show the robot path and the end-points of the scans used for localization.

Acquiring maps with mobile robots is a challenging problem for a number of reasons:

- The hypothesis space, which is the space of all possible maps, is huge. Since maps are defined over a continuous space, the space of all maps has infinitely many dimensions. Even under discrete approximations, such as the grid approximation that shall be used in this chapter, maps can easily be described 10^5 or more variables. The sheer size of this high-dimensional space makes it challenging to calculate full posteriors over maps; hence, the Bayes filtering approach that worked well for localization is inapplicable to the problem of learning maps, at least in its naive form discussed thus far.

- Learning maps is a "chicken-and-egg" problem, for which reason it is often referred to as the *simultaneous localization and mapping* (SLAM) or *concurrent mapping and localization problem*. First, there is a localization problem. When the robot moves through its environment, it accumulates errors in odometry, making it gradually less certain as to where it is. Methods exist for determining the robot's pose when a map is available, as we have seen in the previous chapter. Second, there is a mapping problem. Constructing a map when the robot's poses are known is also relatively easy—a claim that will be substantiated in this chapter and subsequent chapters. In the absence of both an initial map and exact pose information, however, the robot has to do both: estimating the map and localizing itself relative to this map.

Of course, not all mapping problems are equally hard. The hardness of the mapping problem is the result of a collection of factors, the most important of which are:

- **Size.** The larger the environment relative to the robot's perceptual range, the more difficult it is to acquire a map.

- **Noise in perception and actuation.** If robot sensors and actuators were noise-free, mapping would be a simple problem. The larger the noise, the more difficult the problem.

- **Perceptual ambiguity.** The more frequently different places look alike, the more difficult it is to establish correspondence between different locations traversed at different points in time.

(a)

(b)

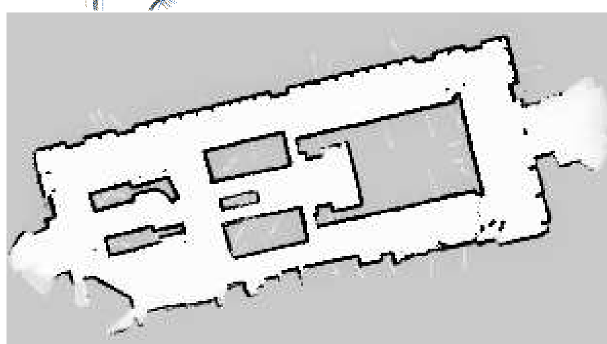


Figure 9.1 (a) Raw range data, position indexed by odometry. (b) Occupancy grid map.

- **Cycles.** Cycles in the environment are particularly difficult to map. If a robot just goes up and down a corridor, it can correct odometry errors incrementally when coming back. Cycles make robots return via different paths, and when closing a cycle the accumulated odometric error can be huge!

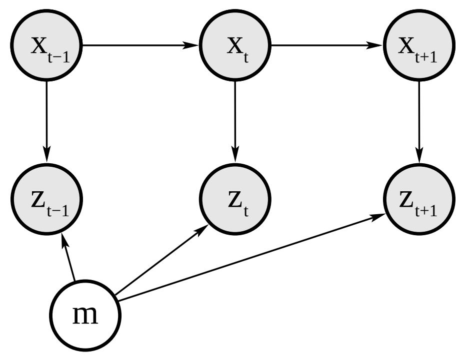


Figure 9.2 Graphical model of mapping with known poses. The shaded variables (poses x and measurements z) are known. The goal of mapping is to recover the map m.

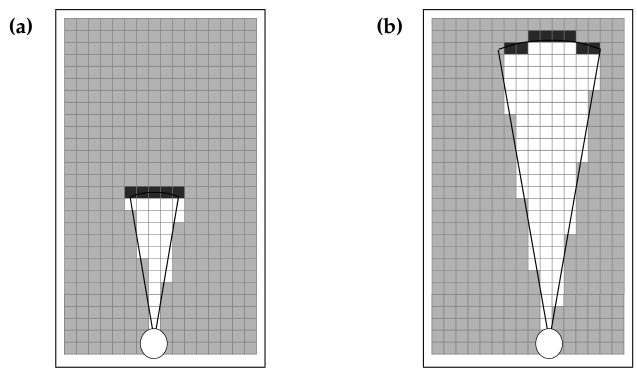


Figure 9.3 Two examples of an inverse measurement model **inverse_range_sensor_model** for two different measurement ranges. The darkness of each grid cell corresponds to the likelihood of occupancy. This model is somewhat simplistic; in contemporary implementations the occupancy probabilities are usually weaker at the border of the measurement cone.

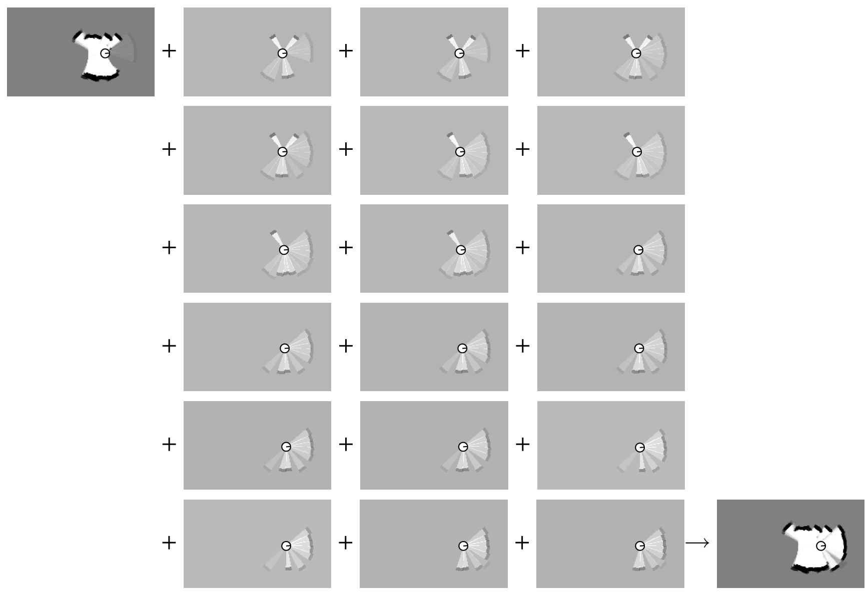


Figure 9.4 Incremental learning of an occupancy grid map using ultra-sound data in a corridor environment. The upper left image shows the initial map and the lower right image contains the resulting map. The maps in columns 2 to 4 are the local maps built from an inverse sensor model. Measurements beyond a 2.5m radius have not been considered. Each cone has an opening angle of 15 degrees. Images courtesy of Cyrill Stachniss, University of Freiburg.

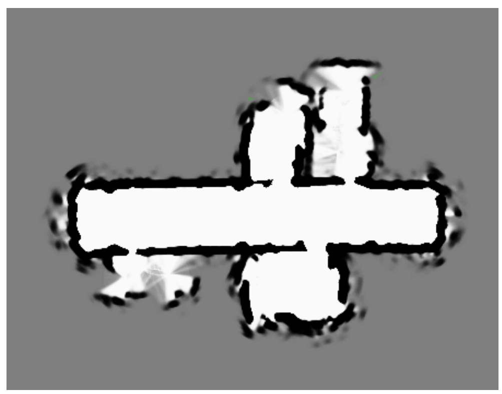


Figure 9.5 Occupancy probability map of an office environment built from sonar measurements. Courtesy of Cyrill Stachniss, University of Freiburg.

(a)

(b)

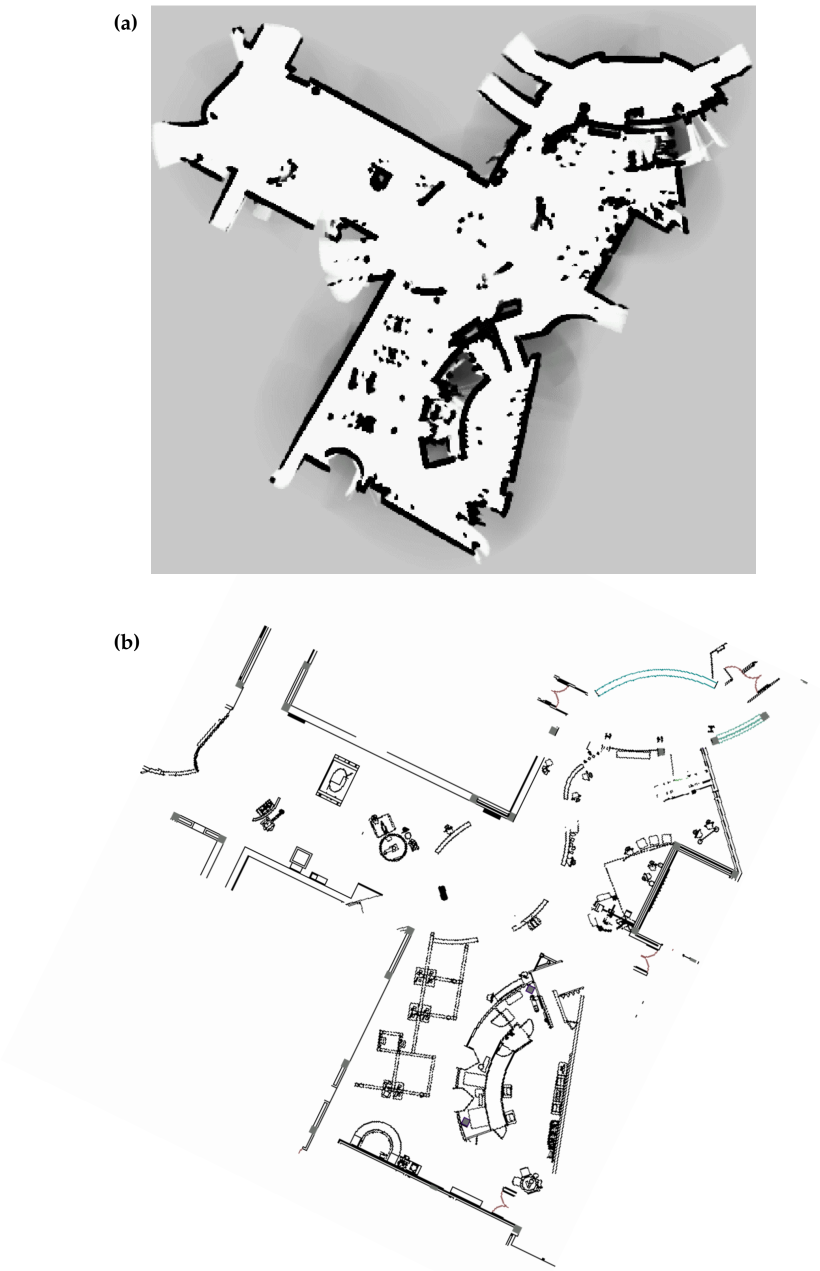


Figure 9.6 (a) Occupancy grid map and (b) architectural blue-print of a large open exhibit space. Notice that the blue-print is inaccurate in certain places.

(a)

(b)

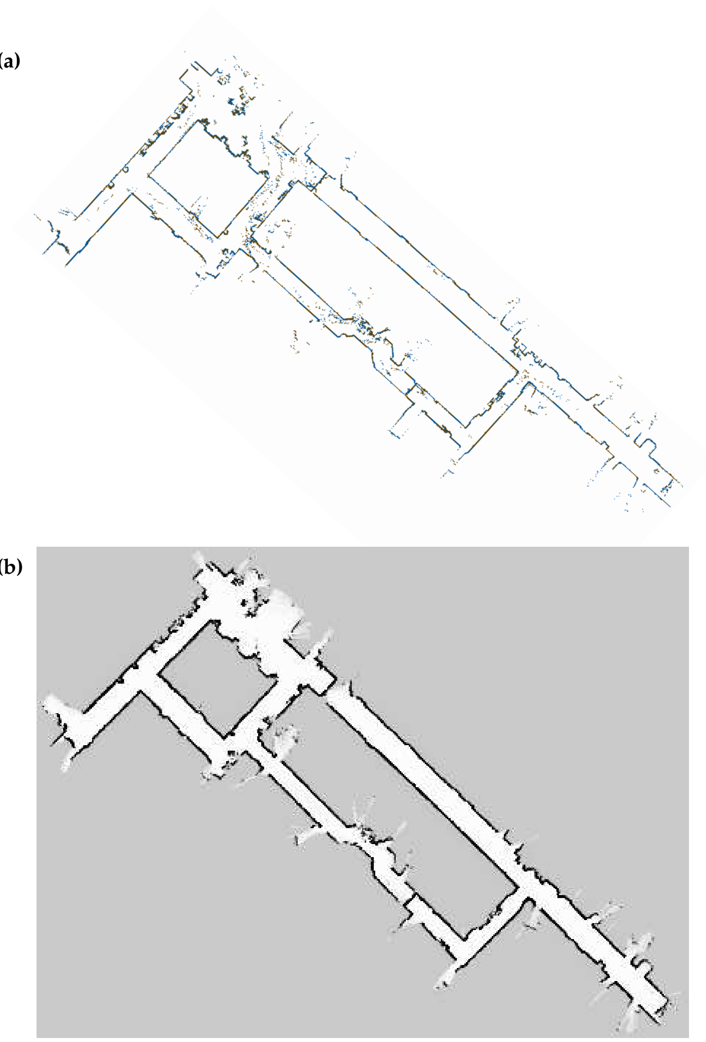


Figure 9.7 (a) Raw laser range data with corrected pose information. Each dot corresponds to a detection of an obstacle. Most obstacles are static (walls etc.), but some were dynamic, since people walked near the robot during data acquisition. (b) Occupancy grid map built from the data. The gray-scale indicates the posterior probability: Black corresponds to occupied with high certainty, and white to free with high certainty. The gray background color represents the prior. Figure (a) courtesy of Steffen Gutmann.

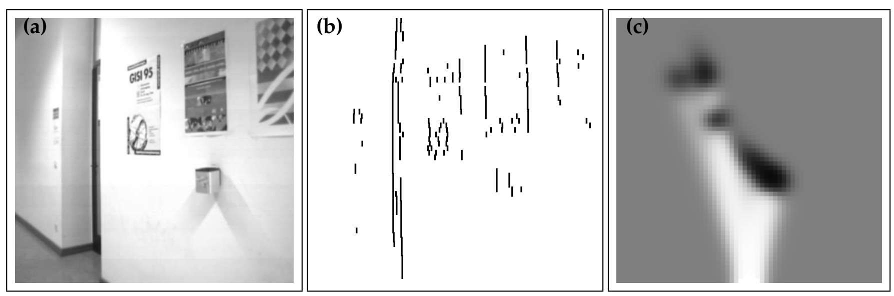


Figure 9.8 Estimation of occupancy maps using stereo vision: (a) camera image, (b) sparse disparity map, (c) occupancy map by projecting the disparity image onto the 2-D plane and convolving the result with a Gaussian. Images courtesy of Thorsten Fröhlinghaus.

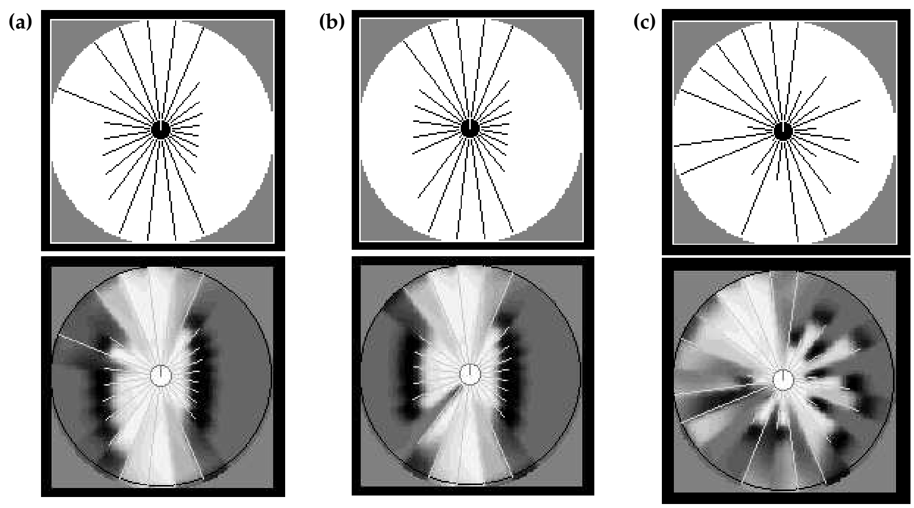


Figure 9.9 Inverse sensor model learned from data: Three sample sonar scans (top row) and local occupancy maps (bottom row), as generated by the neural network. Bright regions indicate free-space, and dark regions indicate walls and obstacles (enlarged by a robot diameter).

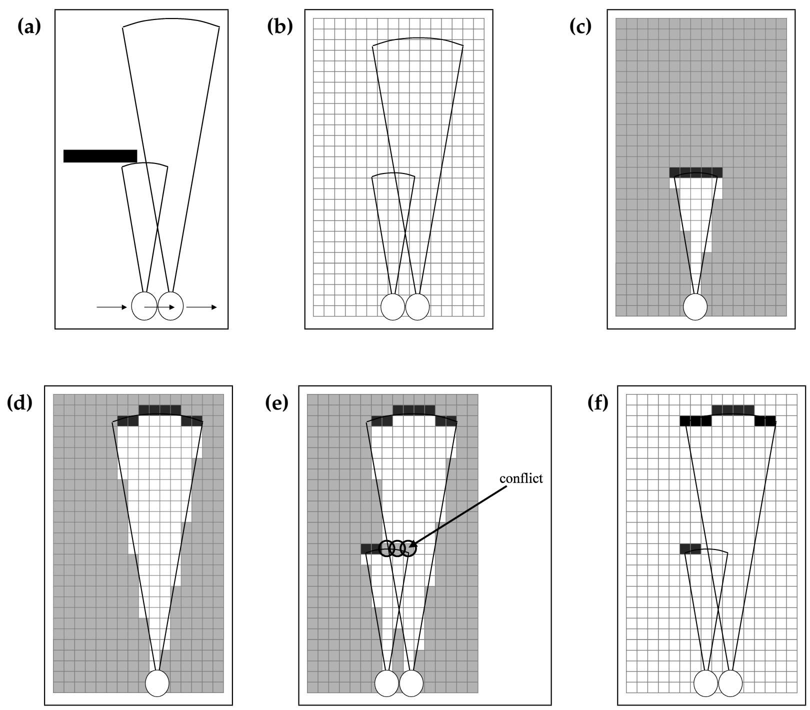


Figure 9.10 The problem with the standard occupancy grid mapping algorithm in Chapter **??**: For the environment shown in Figure (a), a passing robot might receive the (noise-free) measurement shown in (b). The factorial approach maps these beams into probabilistic maps separately for each grid cell and each beam, as shown in (c) and (d). Combining both interpretations yields the map shown in (e). Obviously, there is a conflict in the overlap region, indicated by the circles in (e). The interesting insight is: There exist maps, such as the one in diagram (f), that perfectly explain the sensor measurement without any such conflict. For a sensor reading to be explained, it suffices to assume an obstacle *somewhere* in the cone of a measurement, and not everywhere.

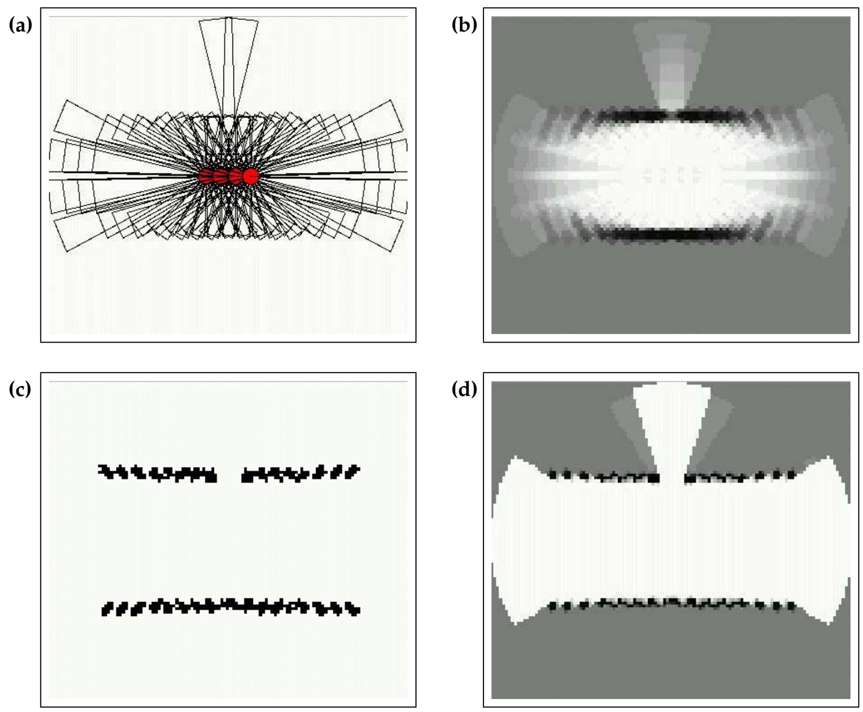


Figure 9.11 (a) Sonar range measurements from a noise-free simulation. (b) Results of the standard occupancy mapper, lacking the open door. (c) A maximum a posterior map. (d) The residual uncertainty in this map, obtained by measuring the sensitivity of the map likelihood function with respect to individual grid cells. This map clearly shows the door, and it also contains flatter walls at both ends.

Figure 9.12 Mobile indoor robot of the type RWI B21, with 24 sonar sensors mounted on a circular array around the robot.

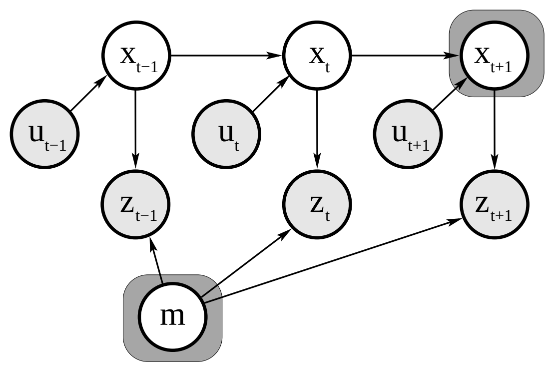


Figure 10.1 Graphical model of the online SLAM problem. The goal of online SLAM is to estimate a posterior over the current robot pose along with the map.

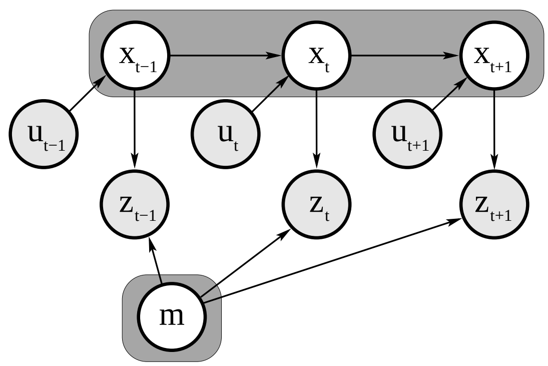


Figure 10.2 Graphical model of the full SLAM problem. Here, we compute a joint posterior over the whole path of the robot and the map.

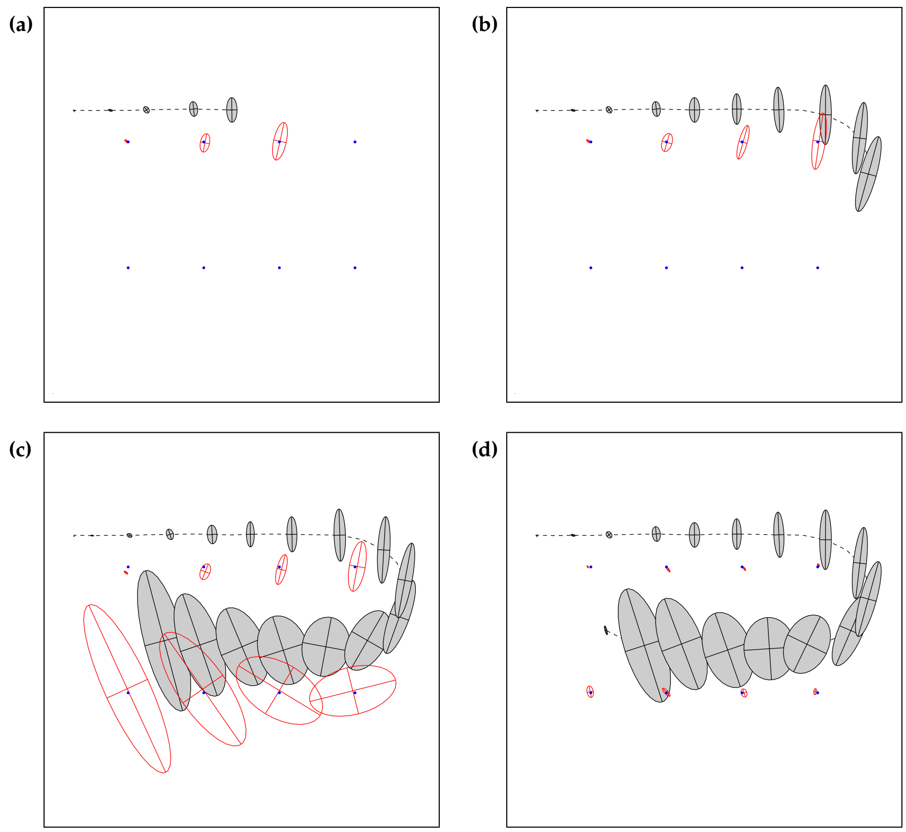


Figure 10.3 EKF applied to the online SLAM problem. The robot's path is a dotted line, and its estimates of its own position are shaded ellipses. Eight distinguishable landmarks of unknown location are shown as small dots, and their location estimates are shown as white ellipses. In (a)–(c) the robot's positional uncertainty is increasing, as is its uncertainty about the landmarks it encounters. In (d) the robot senses the first landmark again, and the uncertainty of *all* landmarks decreases, as does the uncertainty of its current pose. Image courtesy of Michael Montemerlo, Stanford University.

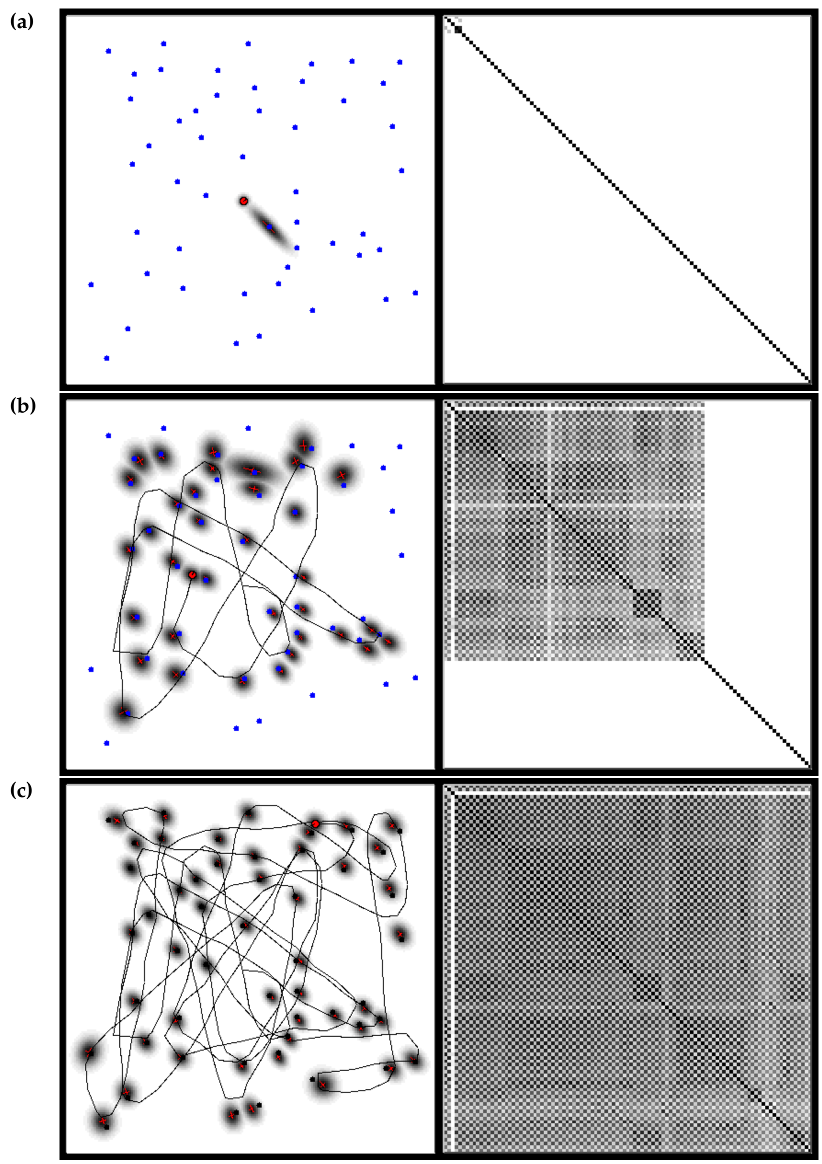


Figure 10.4 EKF SLAM with known data association in a simulated environment. The map is shown on the left, with the gray-level corresponding to the uncertainty of each landmark. The matrix on the right is the correlation matrix, which is the normalized covariance matrix of the posterior estimate. After some time, all x- and all y-coordinate estimates become fully correlated.

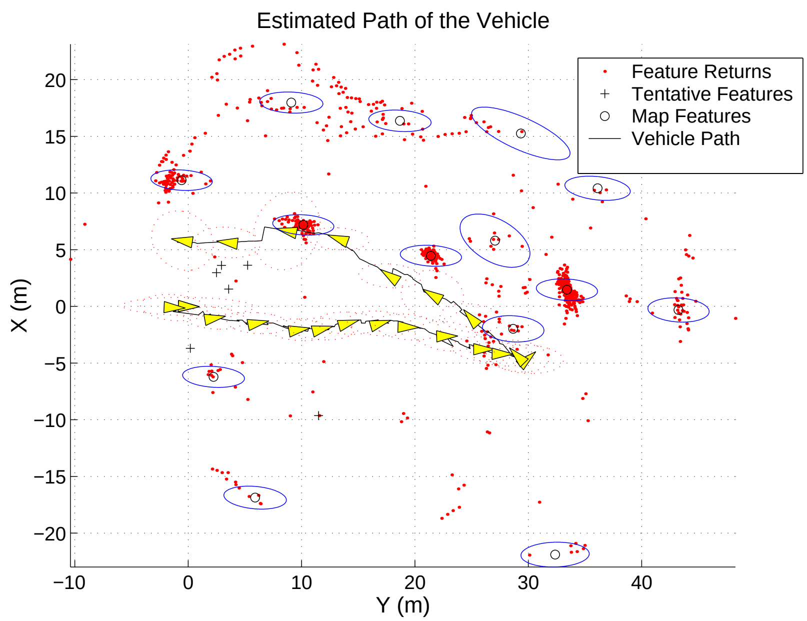


Figure 10.5 Example of Kalman filter estimation of the map and the vehicle pose. Image courtesy of Stefan Williams and Hugh Durrant-Whyte, Australian Centre for Field Robotics.

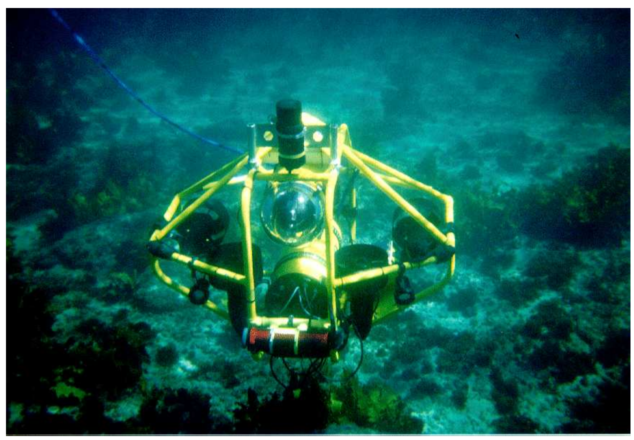


Figure 10.6 Underwater vehicle Oberon, developed at the University of Sydney. Image courtesy of Stefan Williams and Hugh Durrant-Whyte, Australian Centre for Field Robotics.

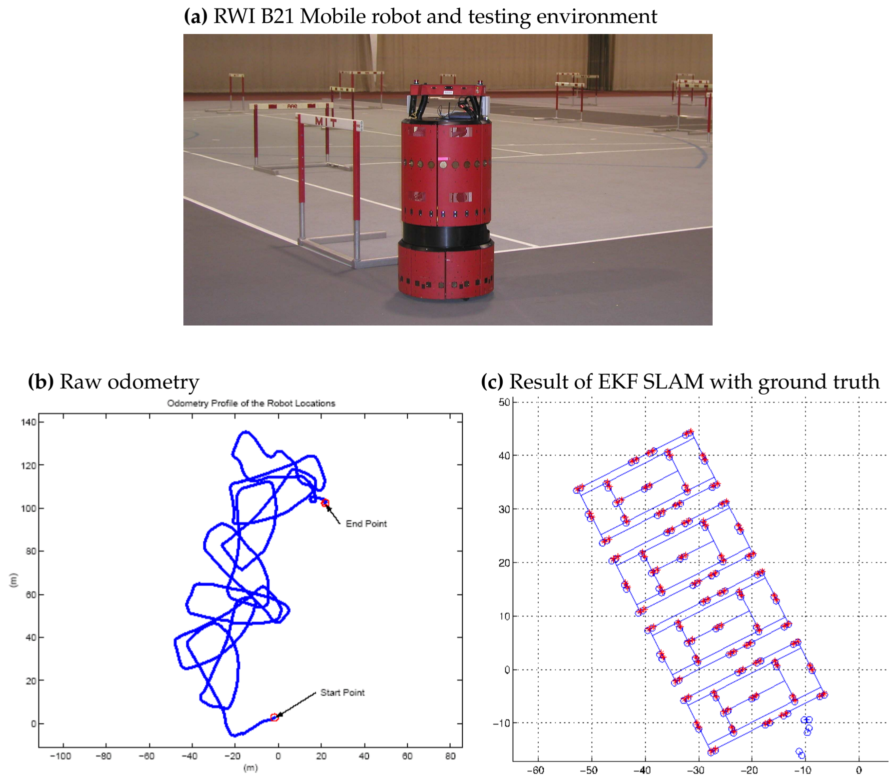
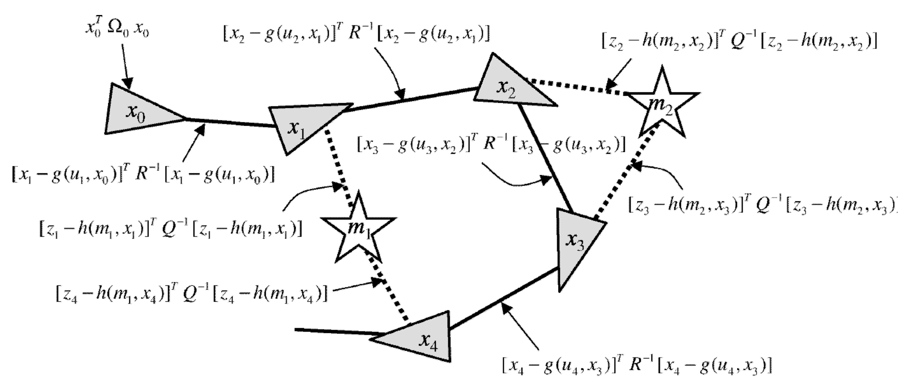


Figure 10.7 (a) The MIT B21 mobile robot in a calibrated testing facility. (b) Raw odometry of the robot, as it is manually driven through the environment. (c) The result of EKF SLAM is a highly accurate map. The image shows the estimated map overlayed on a manually constructed map. All images and results are courtesy of John Leonard and Matthew Walter, MIT.



<u>Sum of all constraints:</u>

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

Figure 11.1 GraphSLAM illustration, with 4 poses and two map features. Nodes in the graphs are robot poses and feature locations. The graph is populated by two types of edges: Solid edges link consecutive robot poses, and dashed edges link poses with features sensed while the robot assumes that pose. Each link in GraphSLAM is a non-linear quadratic constraint. Motion constraints integrate the motion model; measurement constraints the measurement model. The target function of GraphSLAM is sum of these constraints. Minimizing it yields the most likely map and the most likely robot path.

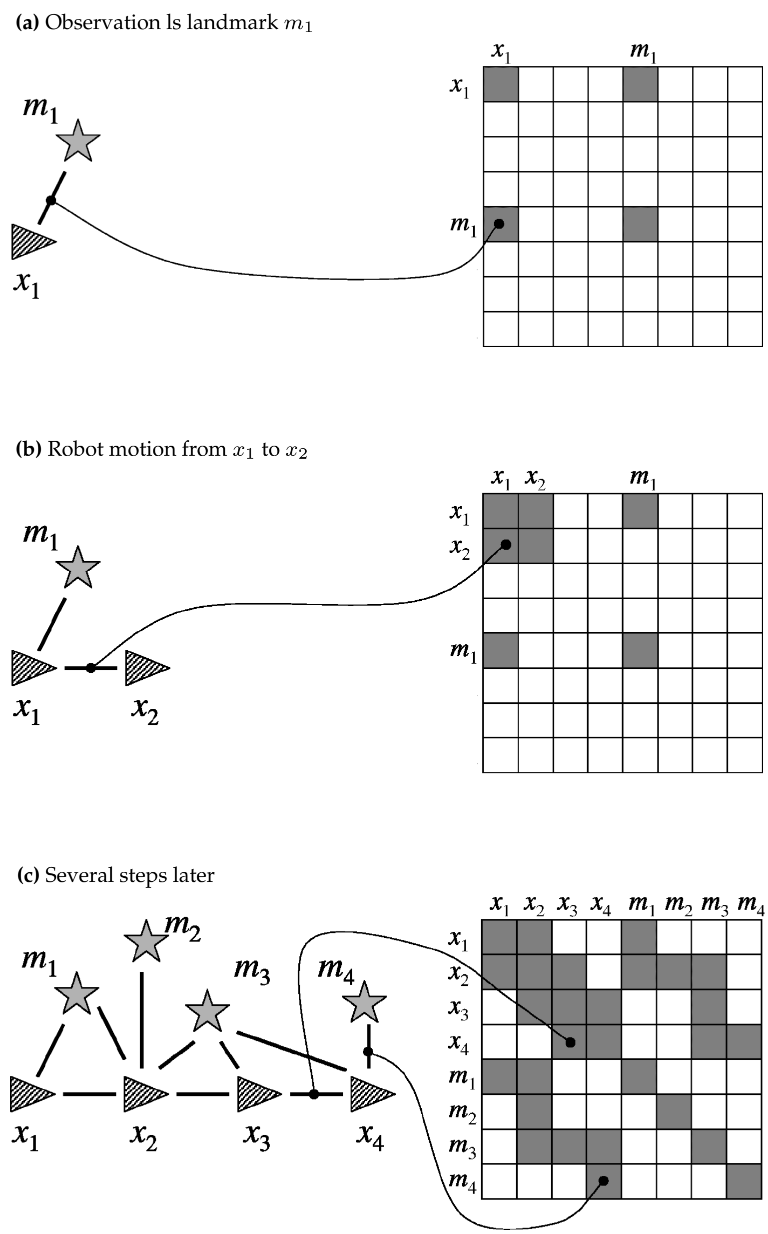
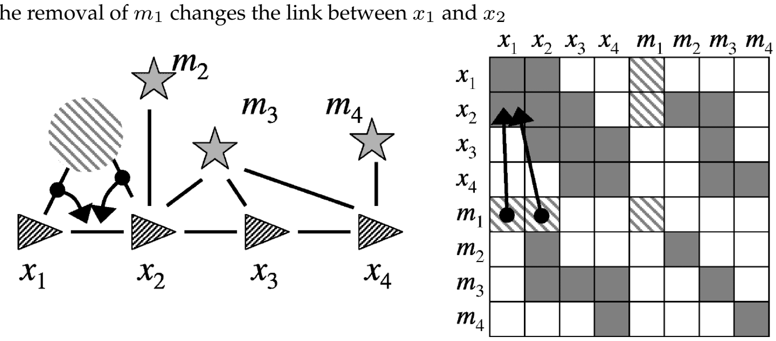
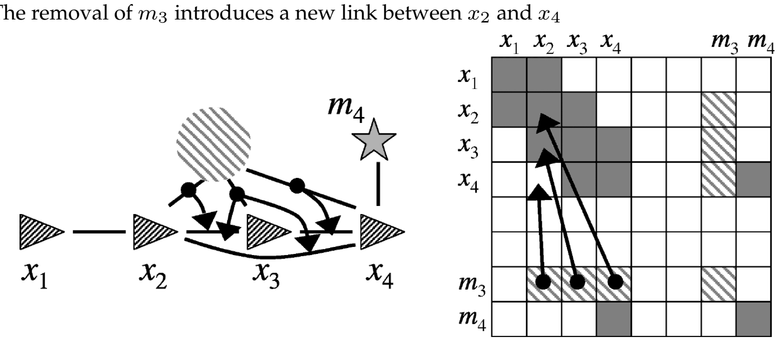


Figure 11.2 Illustration of the acquisition of the information matrix in GraphSLAM. The left diagram shows the dependence graph, the right the information matrix.

(a) The removal of m_1 changes the link between x_1 and x_2



(b) The removal of m_3 introduces a new link between x_2 and x_4



(c) Final Result after removing all map features

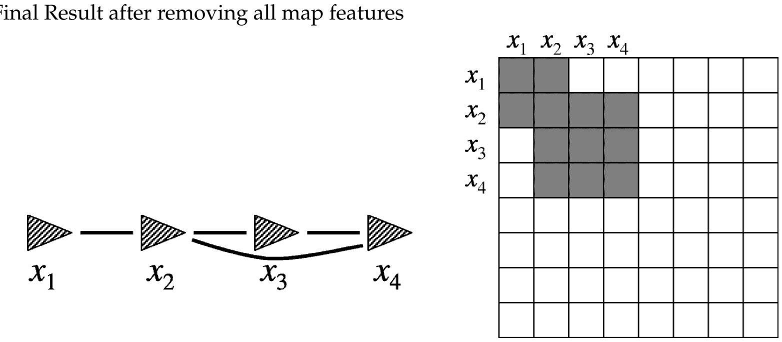


Figure 11.3 Reducing the graph in GraphSLAM: Arcs are removed to yield a network of links that only connect robot poses.

Figure 11.4 The Groundhog robot is a 1,500 pound custom-built vehicle equipped with onboard computing, laser range sensing, gas and sinkage sensors, and video recording equipment. The robot has been built to map abandoned mines.

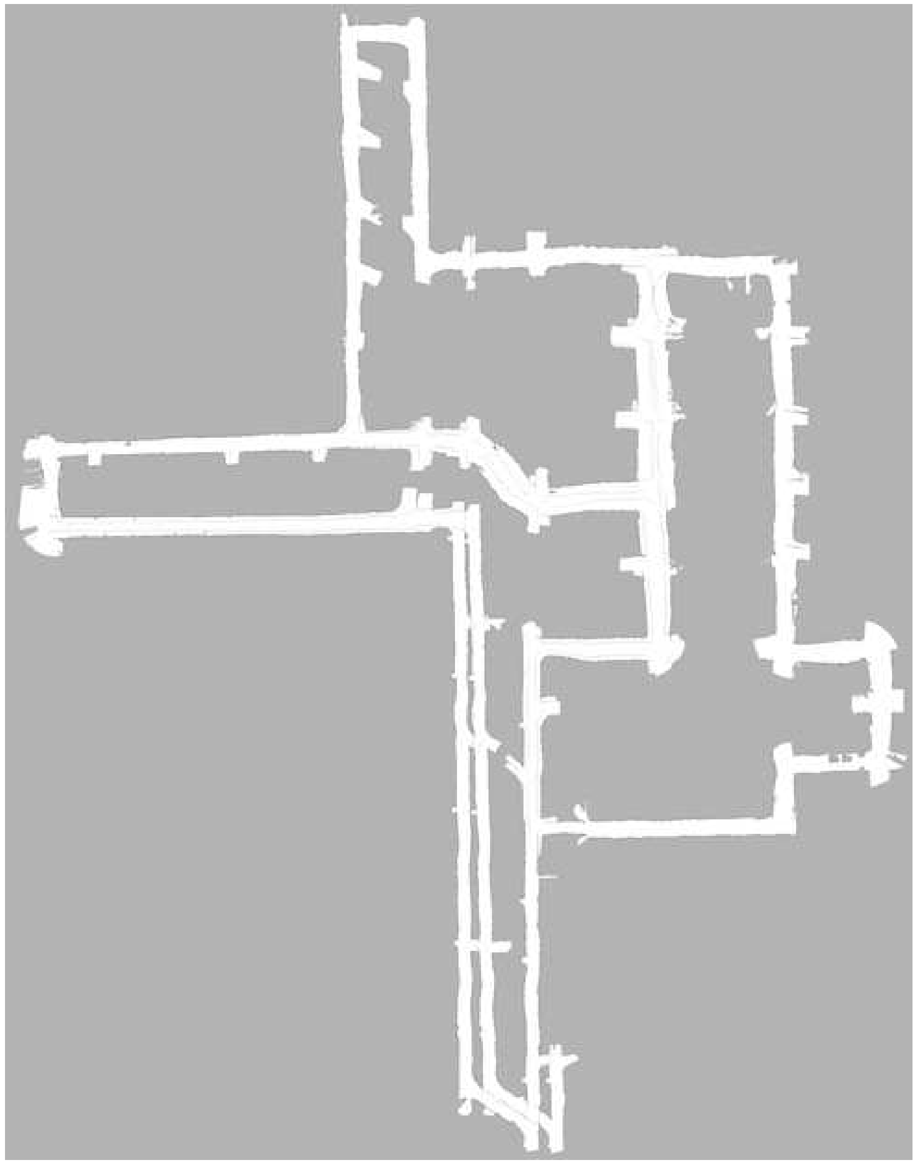


Figure 11.5 Map of a mine, acquired by pairwise scan matching. The diameter of this environment is approximately 250 meters. The map is obviously inconsistent, in that several hallways show up more than once. Image courtesy of Dirk Hähnel, University of Freiburg.

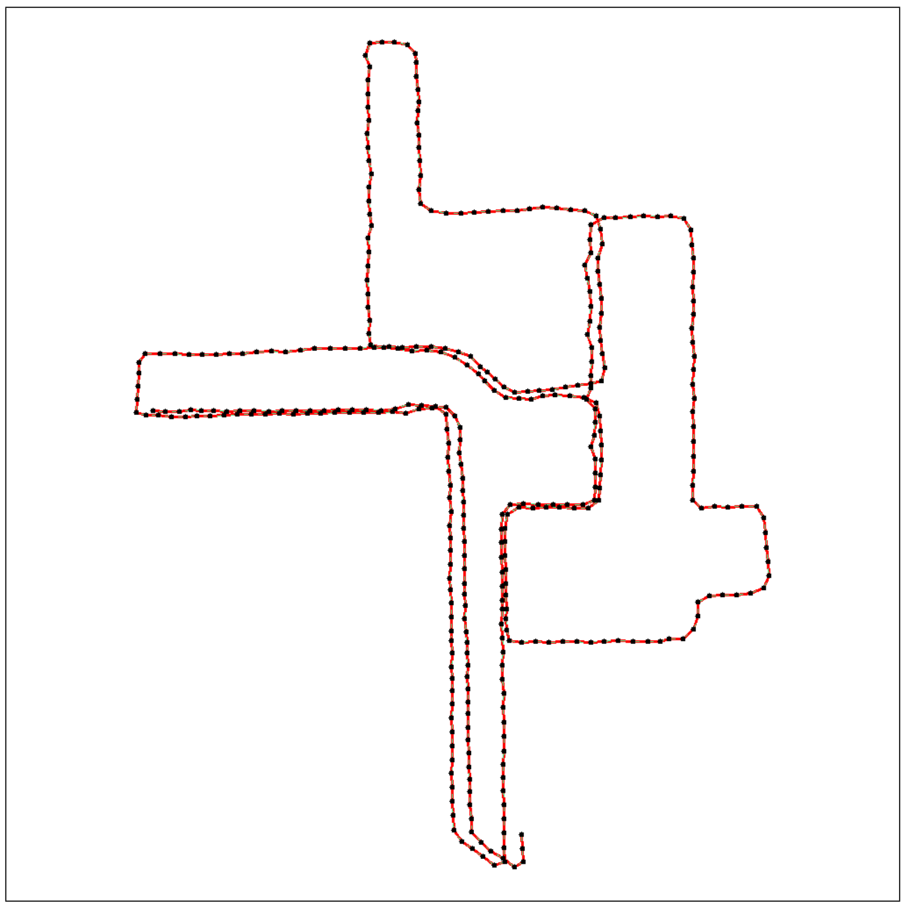


Figure 11.6 Mine map skeleton, visualizing the local maps.

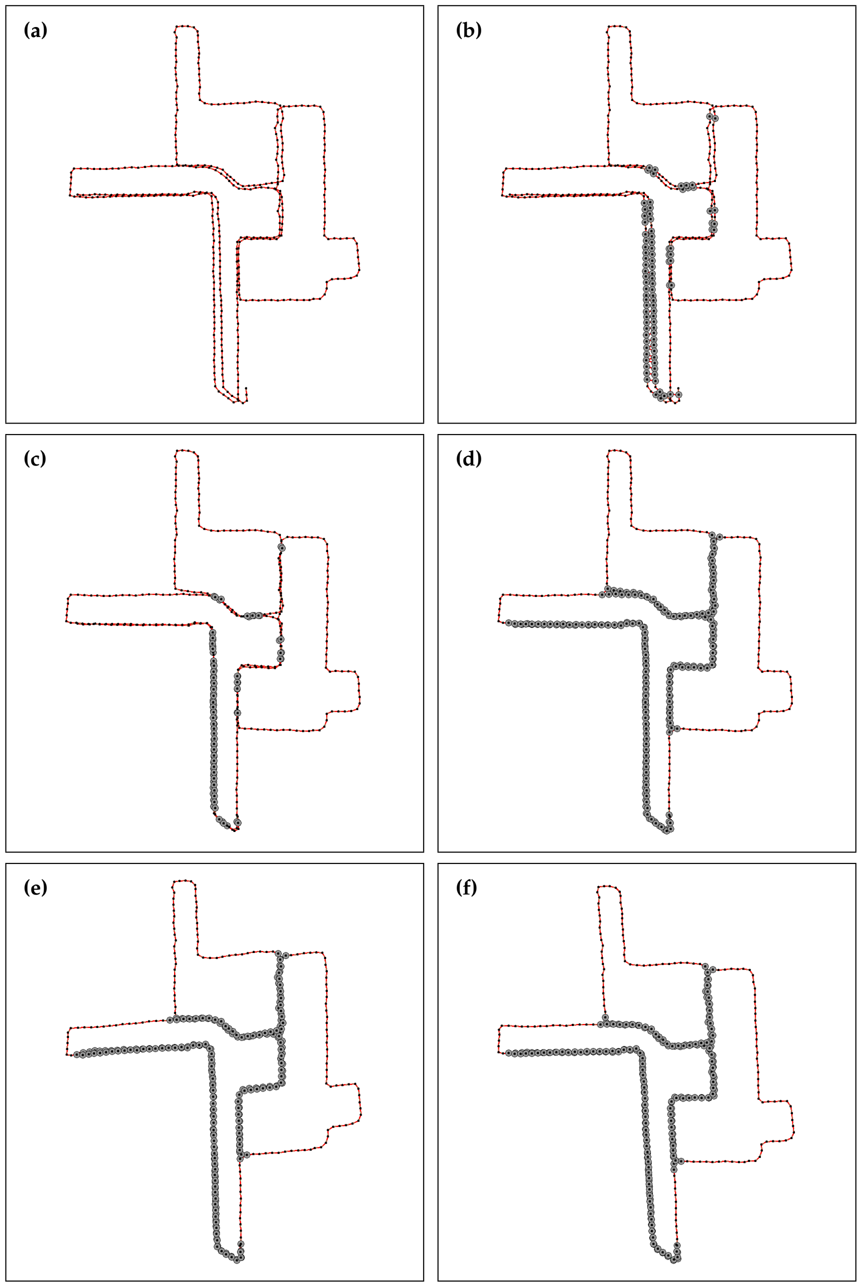


Figure 11.7 Data association search. See text.

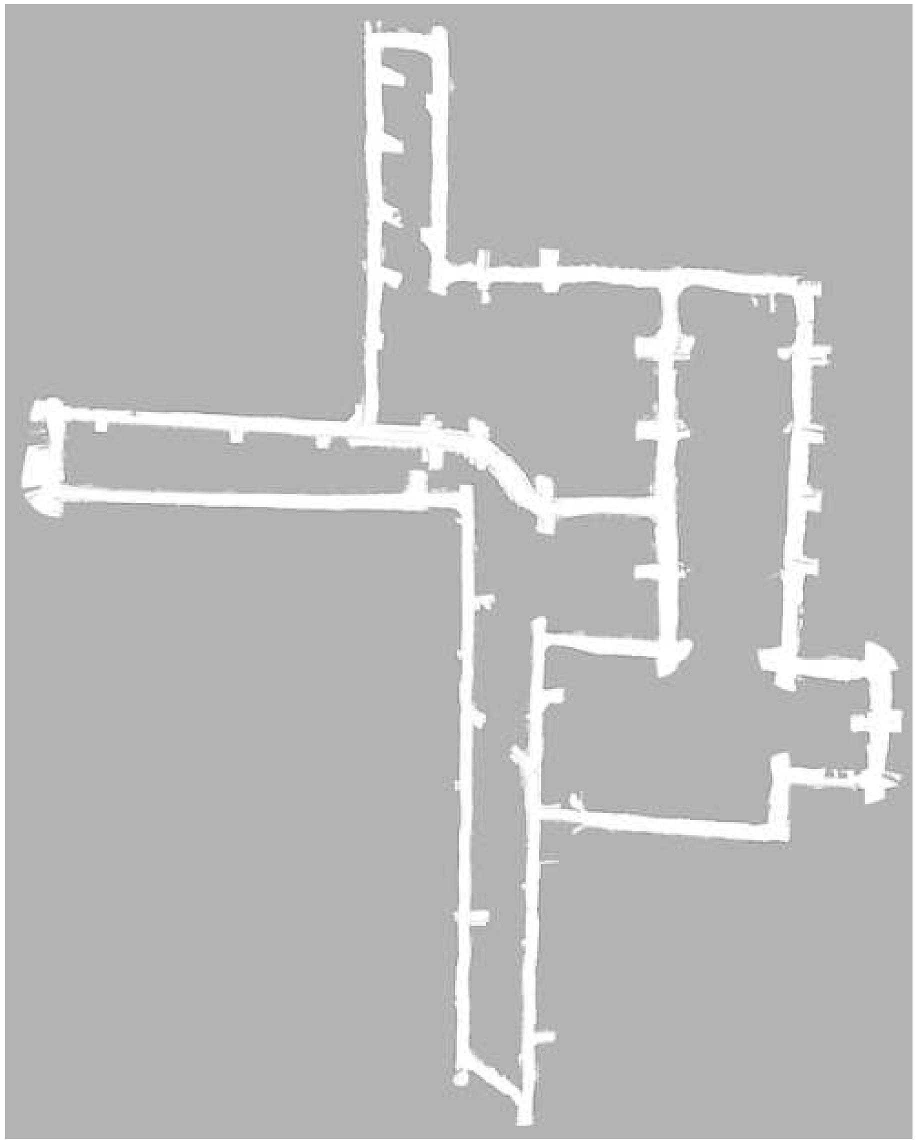


Figure 11.8 Final map, after optimizing for data associations. Image courtesy of Dirk Hähnel, University of Freiburg.

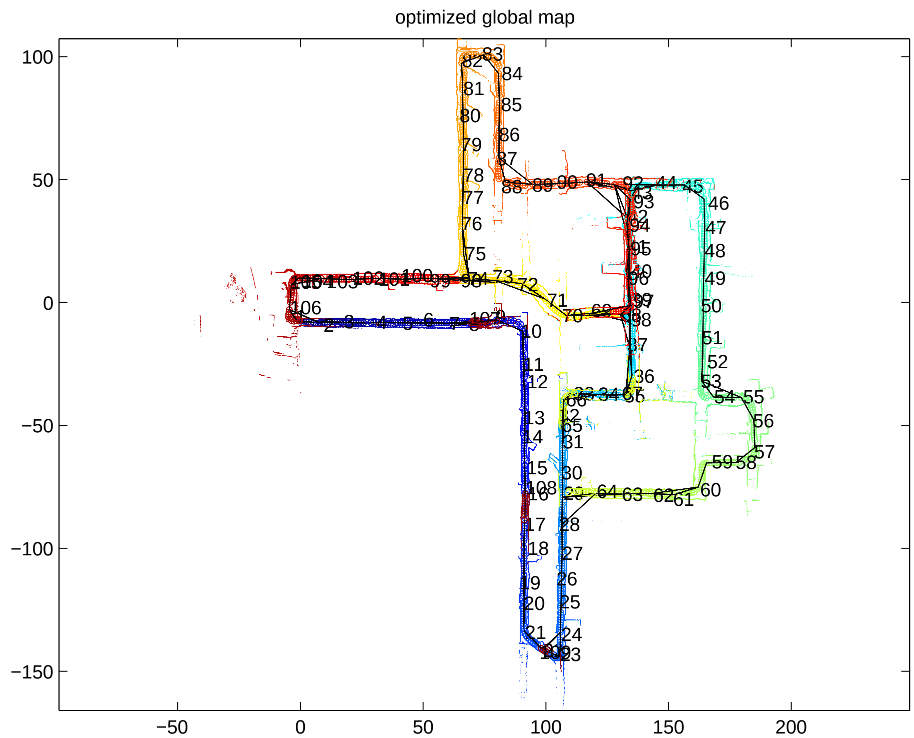


Figure 11.9 Mine map generated by the *Atlas* SLAM algorithm by **?**. Image courtesy of Michael Bosse, Paul Newman, John Leonard, and Seth Teller, MIT.

Figure 11.10 (a) A 3-D map of Stanford's campus. (b) The robot used for acquiring this data is based on a Segway RMP platform, whose development was funded by the DARPA MARS program. Image courtesy of Michael Montemerlo, Stanford University. *(Turn this page 90 degrees to the right to view this figure.)*

Figure 11.11 2-D slice through the Stanford campus map (a) before and (b) after alignment using conjugate gradient. Such an optimization takes only a few seconds with the conjugate gradient method applied to the least square formulation of GraphSLAM. Images courtesy of Michael Montemerlo, Stanford University.

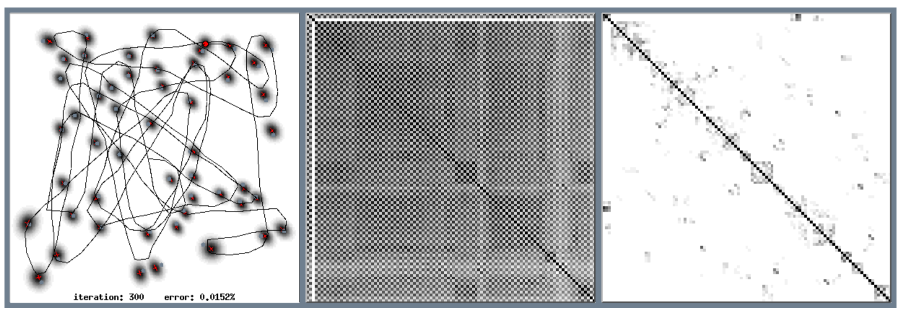


Figure 12.1 Motivation for using an information filter for online SLAM. Left: Simulated robot run with 50 landmarks. Center: The correlation matrix of an EKF, which shows strong correlations between any two landmarks' coordinates. Right: The normalized information matrix of the EKF is naturally sparse. This sparseness leads to a SLAM algorithm that can be updated more efficiently.

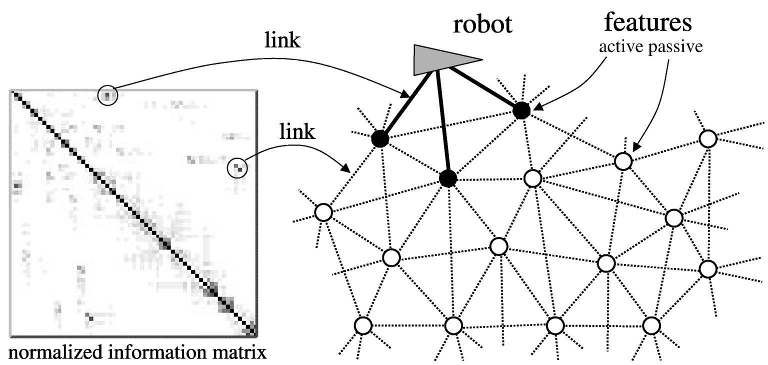


Figure 12.2 Illustration of the network of features generated by our approach. Shown on the left is a sparse information matrix, and on the right a map in which entities are linked whose information matrix element is non-zero. As argued in the text, the fact that not all features are connected is a key structural element of the SLAM problem, and at the heart of our constant time solution.

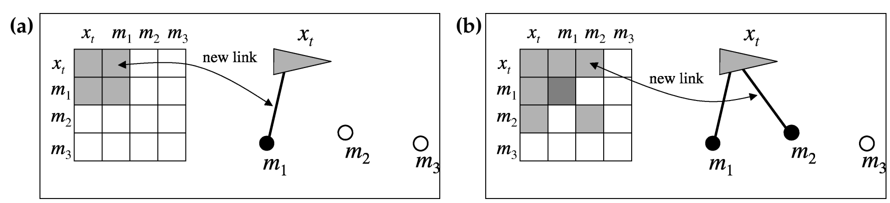


Figure 12.3 The effect of measurements on the information matrix and the associated network of features: (a) Observing m_1 results in a modification of the information matrix elements Ω_{x_t,m_1}. (b) Similarly, observing m_2 affects Ω_{x_t,m_2}.

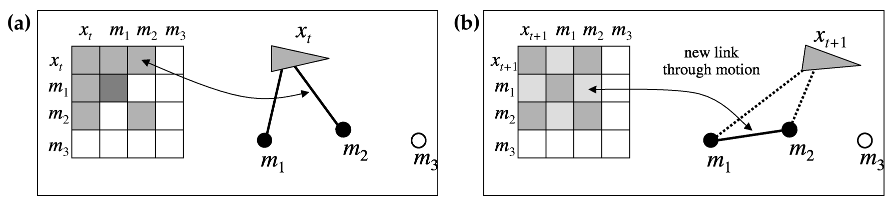


Figure 12.4 The effect of motion on the information matrix and the associated network of features: (a) before motion, and (b) after motion. If motion is nondeterministic, motion updates introduce new links (or reinforce existing links) between any two active features, while weakening the links between the robot and those features. This step introduces links between pairs of features.

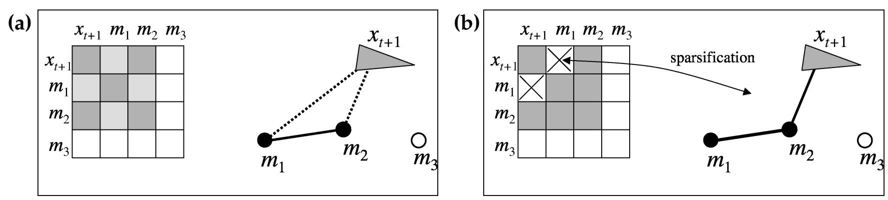


Figure 12.5 Sparsification: A feature is deactivated by eliminating its link to the robot. To compensate for this change in information state, links between active features and/or the robot are also updated. The entire operation can be performed in constant time.

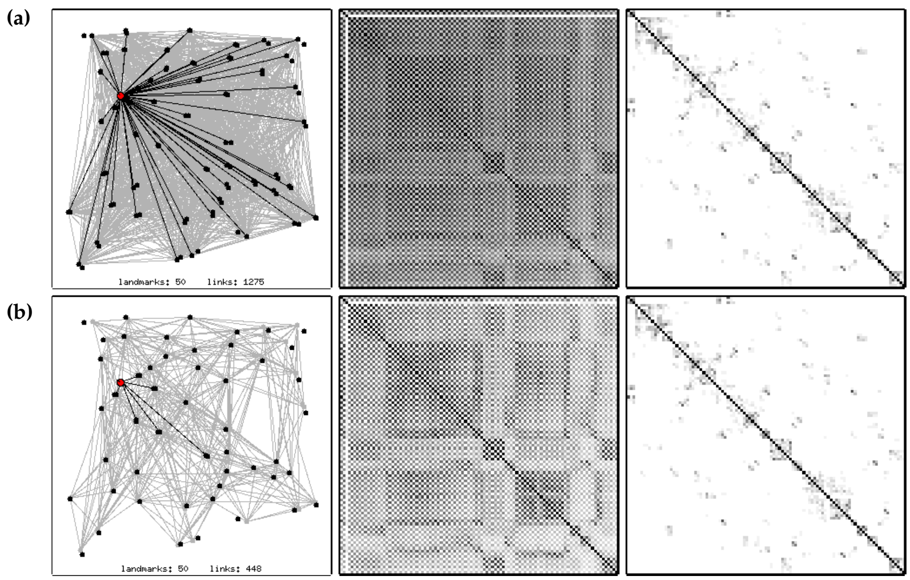


Figure 12.6 Comparison of (a) SEIF without sparsification with (b) SEIF using the sparsification step with 4 active landmarks. The comparison is carried out in a simulated environment with 50 landmarks. In each row, the left panel shows the set of links in the filter, the center panel the correlation matrix, and the right panel the normalized information matrix. Obviously, the sparsified SEIF maintains many fewer links, but its result is less confident as indicated by its less-expressed correlation matrix.

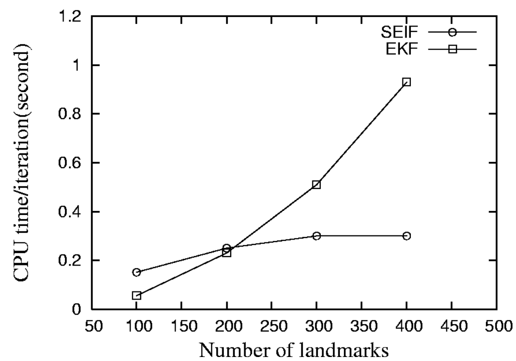


Figure 12.7 The comparison of average CPU time between SEIF and EKF.

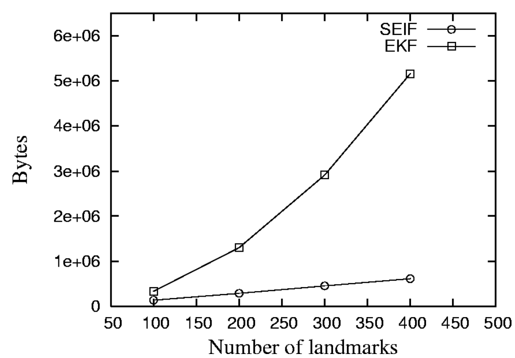


Figure 12.8 The comparison of average memory usage between SEIF and EKF.

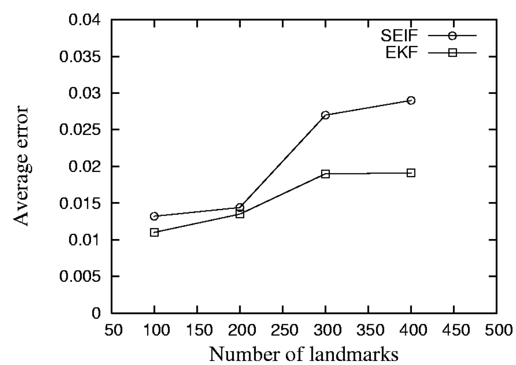


Figure 12.9 The comparison of root mean square distance error between SEIF and EKF.

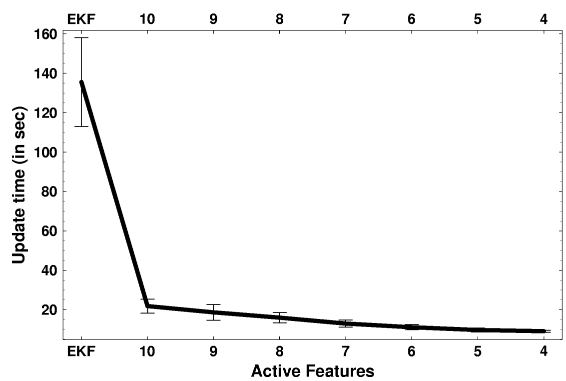


Figure 12.10 The update time of the EKF (leftmost data point only) and the SEIF, for different degrees of sparseness, as induced by a bound on the number of active features as indicated.

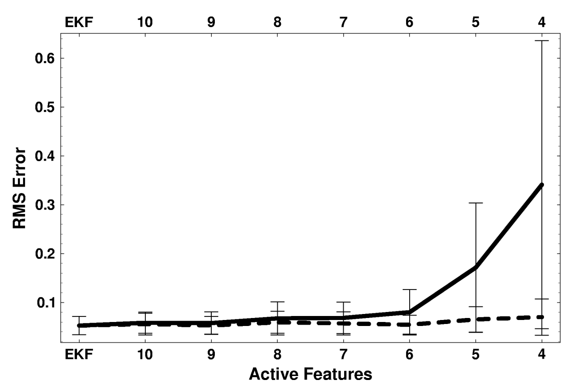


Figure 12.11 The approximation error EKF (leftmost data point only) and SEIF for different degrees of sparseness. In both figures, the map consists of 50 landmarks.

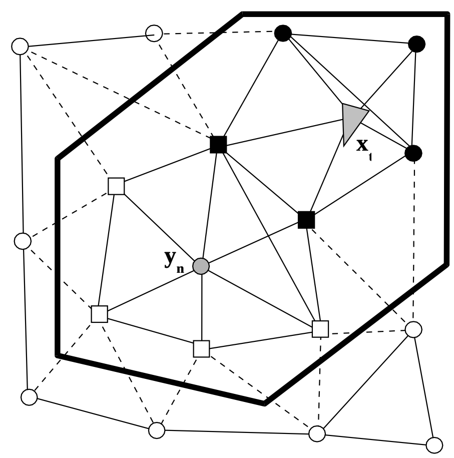


Figure 12.12 The combined Markov blanket of feature y_n and the observed features is usually sufficient for approximating the posterior probability of the feature locations, conditioning away all other features.

Figure 12.13 The vehicle used in our experiments is equipped with a 2-D laser range finder and a differential GPS system. The vehicle's ego-motion is measured by a linear variable differential transformer sensor for the steering, and a wheel-mounted velocity encoder. In the background, the Victoria Park test environment can be seen. Image courtesy of José Guivant and Eduardo Nebot, Australian Centre for Field Robotics.

Figure 12.14 The testing environment: A 350 meter by 350 meter patch in Victoria Park in Sydney. Overlayed is the integrated path from odometry readings. Data and aerial image courtesy of José Guivant and Eduardo Nebot, Australian Centre for Field Robotics; results courtesy of Michael Montemerlo, Stanford University.

Figure 12.15 The path recovered by the SEIF, is correct within ± 1m. Courtesy of Michael Montemerlo, Stanford University.

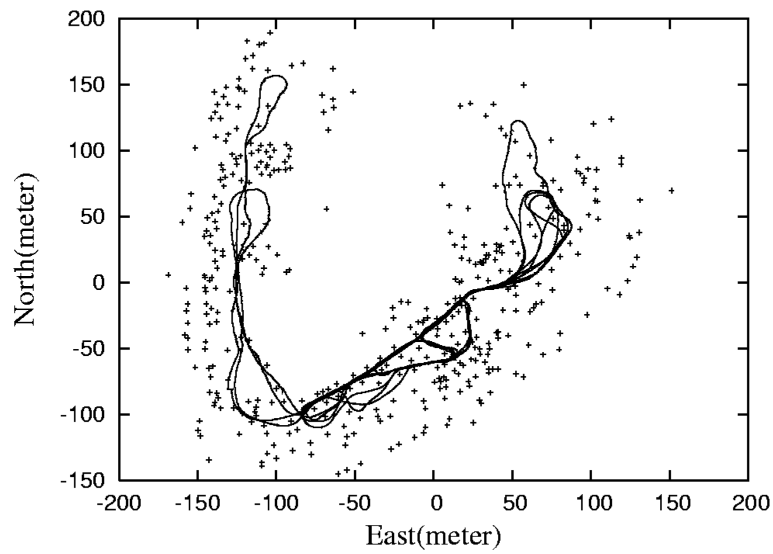


Figure 12.16 Overlay of estimated landmark positions and robot path. Images courtesy of Michael Montemerlo, Stanford University

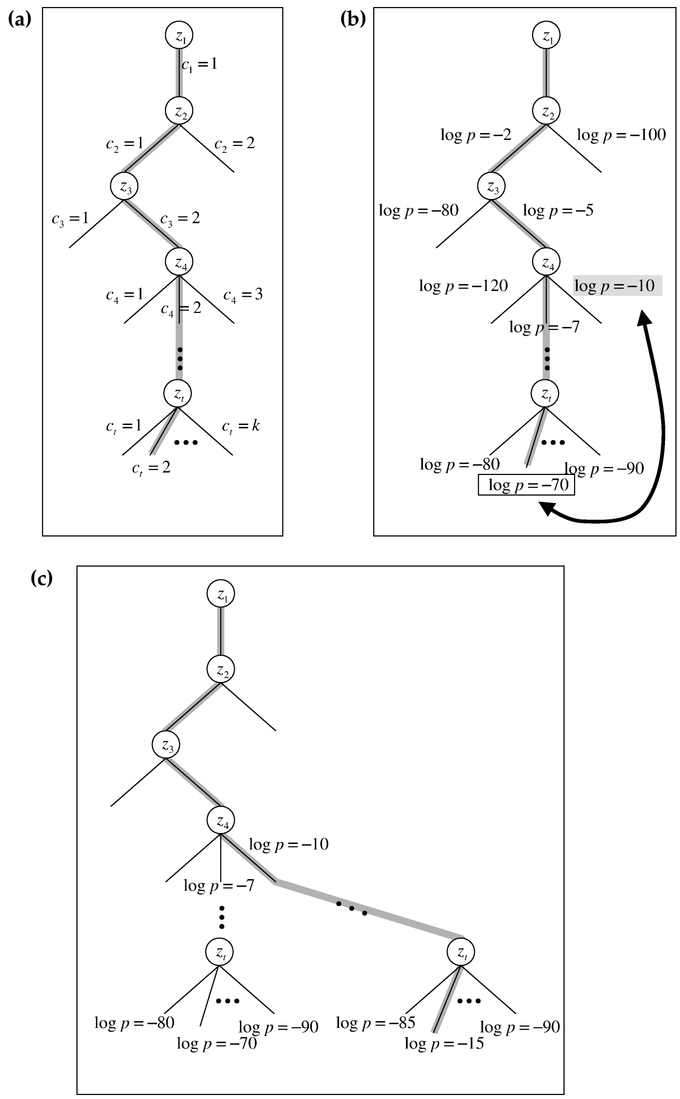


Figure 12.17 (a) The data association tree, whose branching factor grows with the number of landmarks in the map. (b) The tree-based SEIF maintains the log-likelihood for the entire frontier of expanded nodes, enabling it to find alternative paths. (c) Improved path.

(a)

(b)

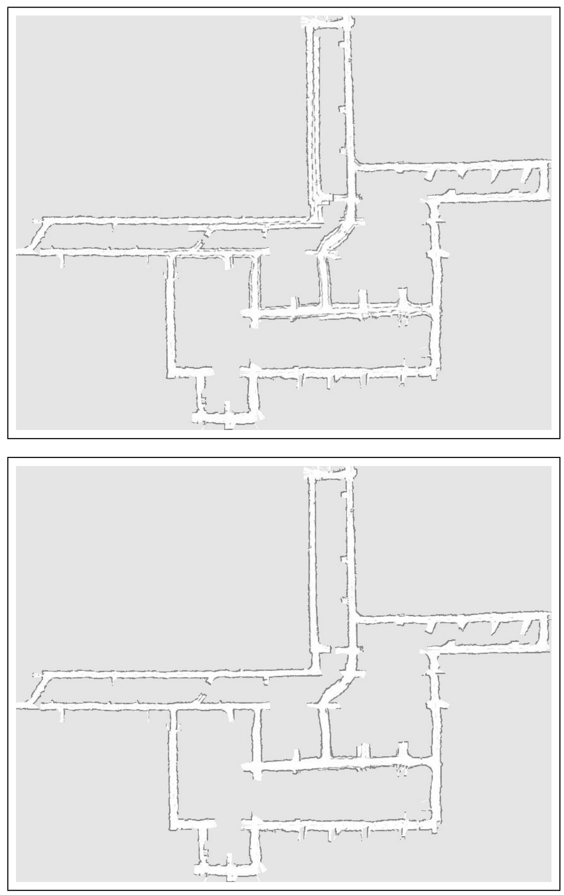


Figure 12.18 (a) Map with incremental ML scan matching and (b) full recursive branch-and-bound data association. Images courtesy of Dirk Hähnel, University of Freiburg.

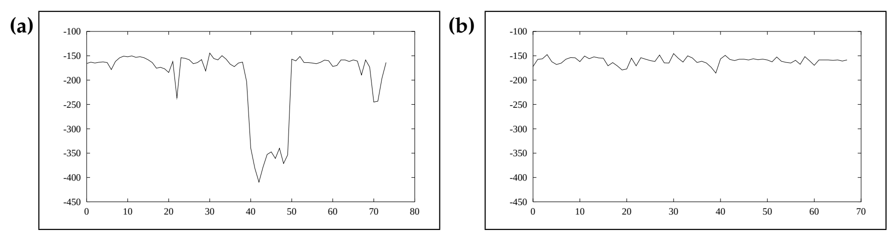


Figure 12.19 (a) Log-likelihood of the actual measurement, as a function of time. The lower likelihood is caused by the wrong assignment. (b) Log-likelihood, when recursively fixing false data association hypotheses through the tree search. The success is manifested by the lack of a distinct dip.

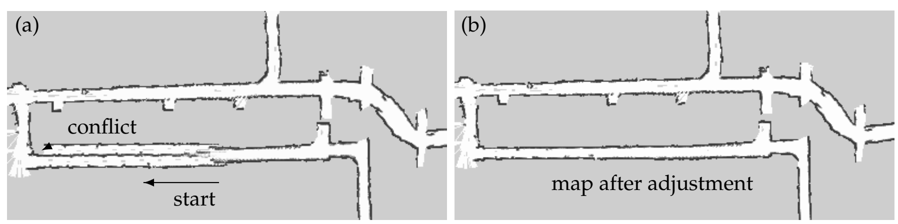


Figure 12.20 Example of the tree-based data association technique: (a) When closing a large loop, the robot first erroneously assumes the existence of a second, parallel hallway. However, this model leads to a gross inconsistency as the robot encounters a corridor at a right angle. At this point, the approach recursively searches for improved data association decisions, arriving on the map shown in diagram (b).

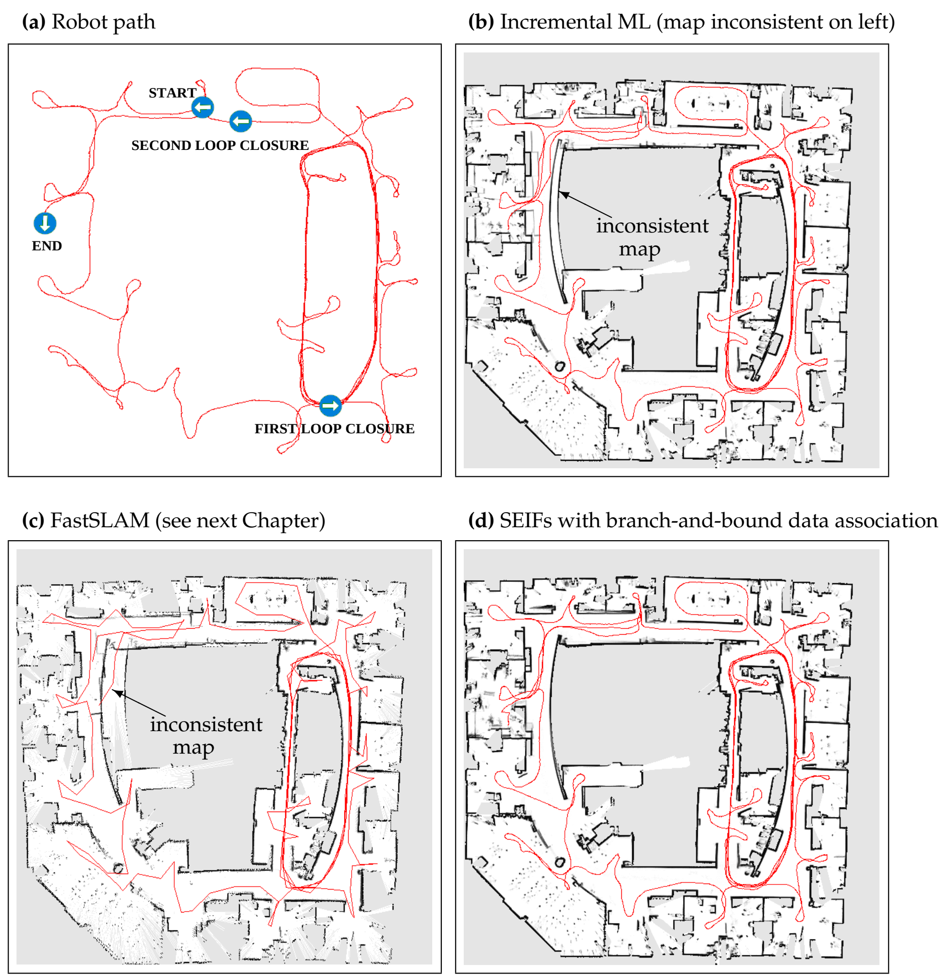


Figure 12.21 (a) Path of the robot. (b) Incremental ML (scan matching) (c) FastSLAM. (d) SEIFs with lazy data association. Image courtesy of Dirk Hähnel, University of Freiburg.

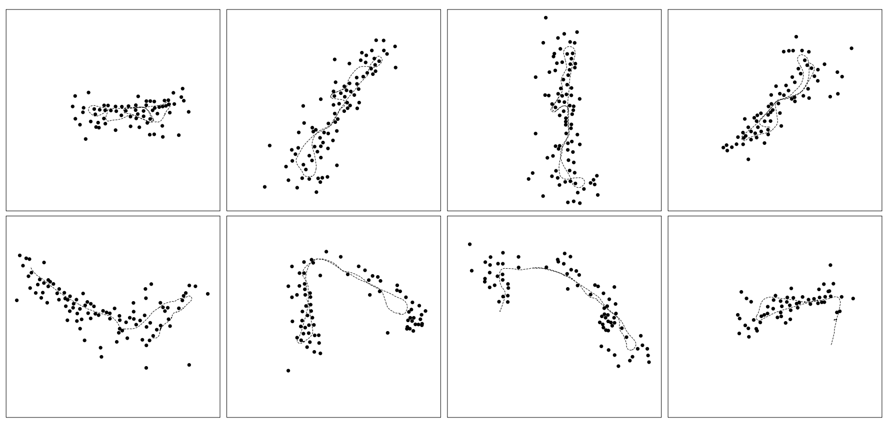


Figure 12.22 Eight local maps obtained by splitting the data into eight sequences.

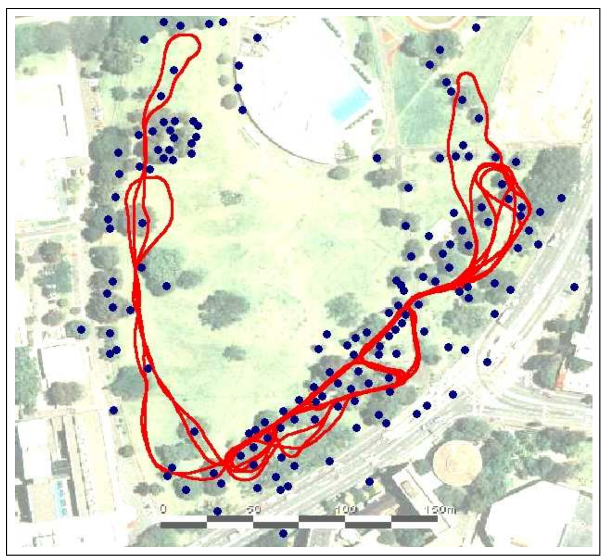


Figure 12.23 A multi-robot SLAM result, obtained using the algorithm described in this chapter. Image courtesy of Yufeng Liu.

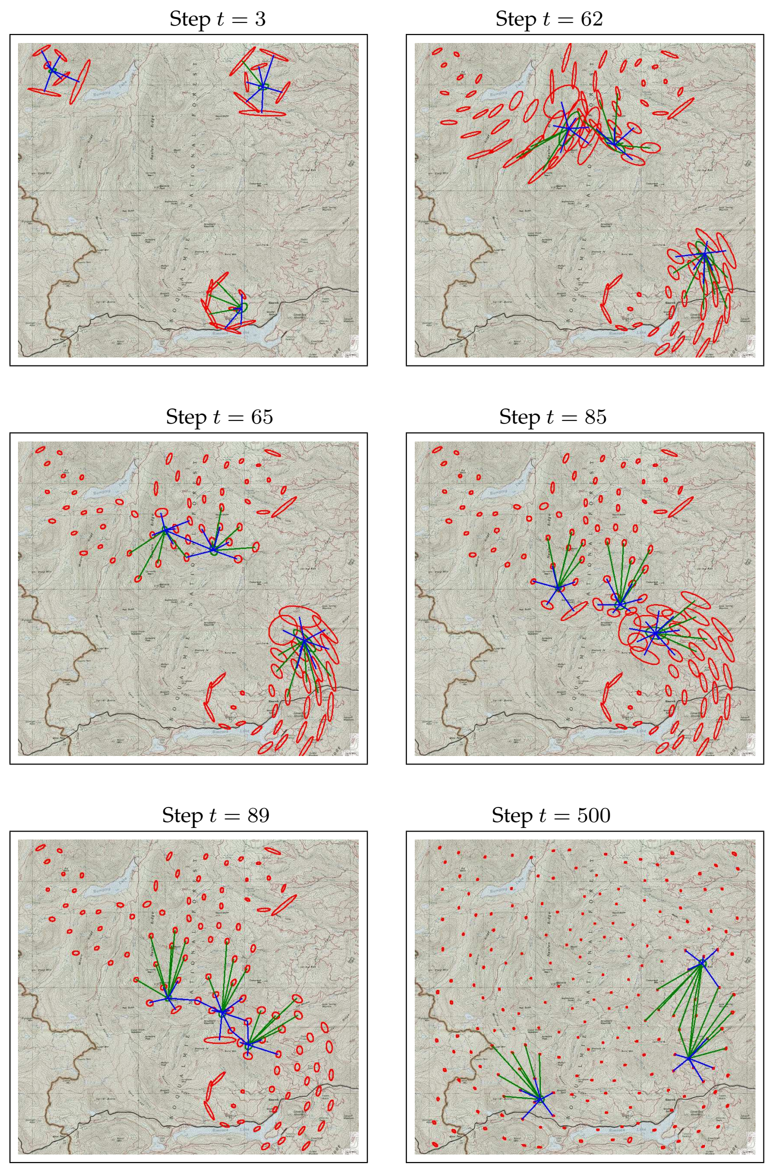


Figure 12.24 Snapshots from our multi-robot SLAM simulation at different points in time. During Steps 62 through 64, vehicle 1 and 2 traverse the same area for the first time; as a result, the uncertainty in their local maps shrinks. Later, in steps 85 through 89, vehicle 2 observes the same landmarks as vehicle 3, with a similar effect on the overall uncertainty. After 500 steps, all landmarks are accurately localized.

	robot path	feature 1	feature 2	…	feature N
Particle $k=1$	$x_{1:t}^{[1]} = \{(x\ \ y\ \ \theta)^T\}_{1:t}^{[1]}$	$\mu_1^{[1]}, \Sigma_1^{[1]}$	$\mu_2^{[1]}, \Sigma_2^{[1]}$	…	$\mu_N^{[1]}, \Sigma_N^{[1]}$
Particle $k=2$	$x_{1:t}^{[2]} = \{(x\ \ y\ \ \theta)^T\}_{1:t}^{[2]}$	$\mu_1^{[2]}, \Sigma_1^{[2]}$	$\mu_2^{[2]}, \Sigma_2^{[2]}$	…	$\mu_N^{[2]}, \Sigma_N^{[2]}$
		$\vdots$			
Particle $k=M$	$x_{1:t}^{[M]} = \{(x\ \ y\ \ \theta)^T\}_{1:t}^{[M]}$	$\mu_1^{[M]}, \Sigma_1^{[M]}$	$\mu_2^{[M]}, \Sigma_2^{[M]}$	…	$\mu_N^{[M]}, \Sigma_N^{[M]}$

Figure 13.1 Particles in FastSLAM are composed of a path estimate and a set of estimators of individual feature locations with associated covariances.

- Do the following M times:
  - **Retrieval.** Retrieve a pose $x_{t-1}^{[k]}$ from the particle set Y_{t-1}.
  - **Prediction.** Sample a new pose $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$.
  - **Measurement update.** For each observed feature z_t^i identify the correspondence j for the measurement z_t^i, and incorporate the measurement z_t^i into the corresponding EKF, by updating the mean $\mu_{j,t}^{[k]}$ and covariance $\Sigma_{j,t}^{[k]}$.
  - **Importance weight.** Calculate the importance weight $w^{[k]}$ for the new particle.
- **Resampling.** Sample, with replacement, M particles, where each particle is sampled with a probability proportional to $w^{[k]}$.

Figure 13.2 The basic steps of the FastSLAM algorithm.

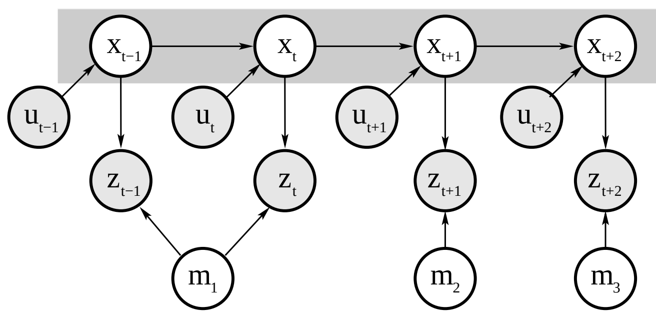


Figure 13.3 The SLAM problem depicted as Bayes network graph. The robot moves from pose x_{t-1} to pose x_{t+2}, driven by a sequence of controls. At each pose x_t it observes a nearby feature in the map $m = \{m_1, m_2, m_3\}$. This graphical network illustrates that the pose variables "separate" the individual features in the map from each other. If the poses are known, there remains no other path involving variables whose value is not known, between any two features in the map. This lack of a path renders the posterior of any two features in the map conditionally independent (given the poses).

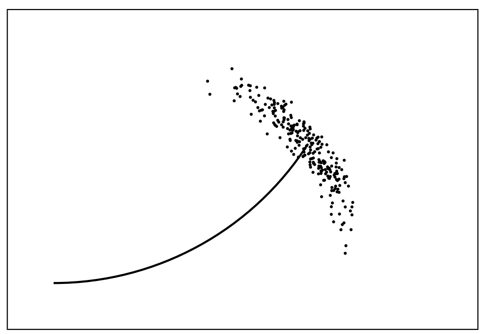


Figure 13.4 Samples drawn from the probabilistic motion model.

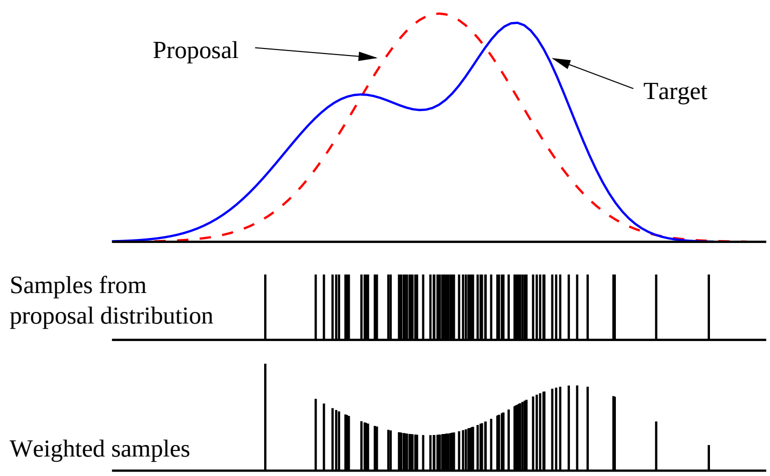


Figure 13.5 Samples cannot be drawn conveniently from the target distribution (shown as a solid line). Instead, the importance sampler draws samples from the proposal distribution (dashed line), which has a simpler form. Below, samples drawn from the proposal distribution are drawn with lengths proportional to their importance weights.

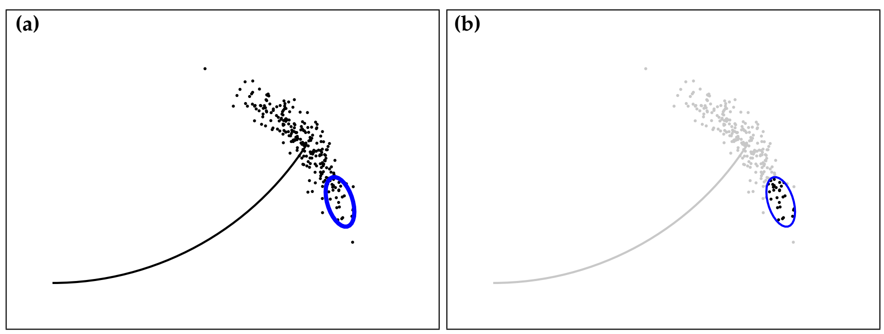


Figure 13.6 Mismatch between proposal and posterior distributions: (a) illustrates the forward samples generated by FastSLAM 1.0, and the posterior induced by the measurement (ellipse). Diagram (b) shows the sample set after the resampling step.

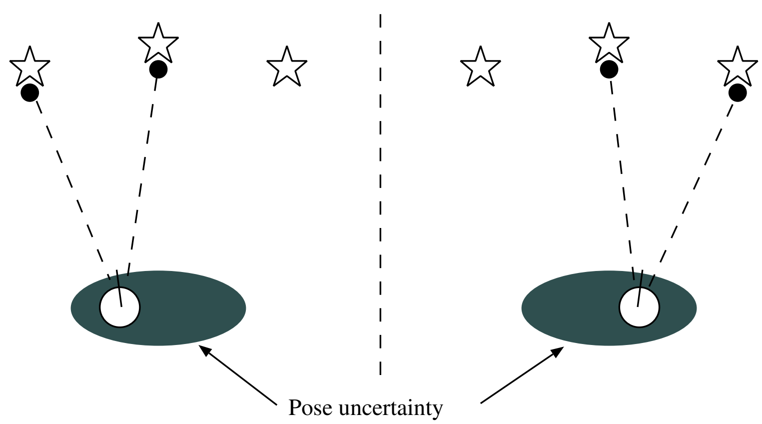


Figure 13.7 The data association problem in SLAM. This figure illustrates that the best data association may vary even within regions of high likelihood for the pose of the robot.

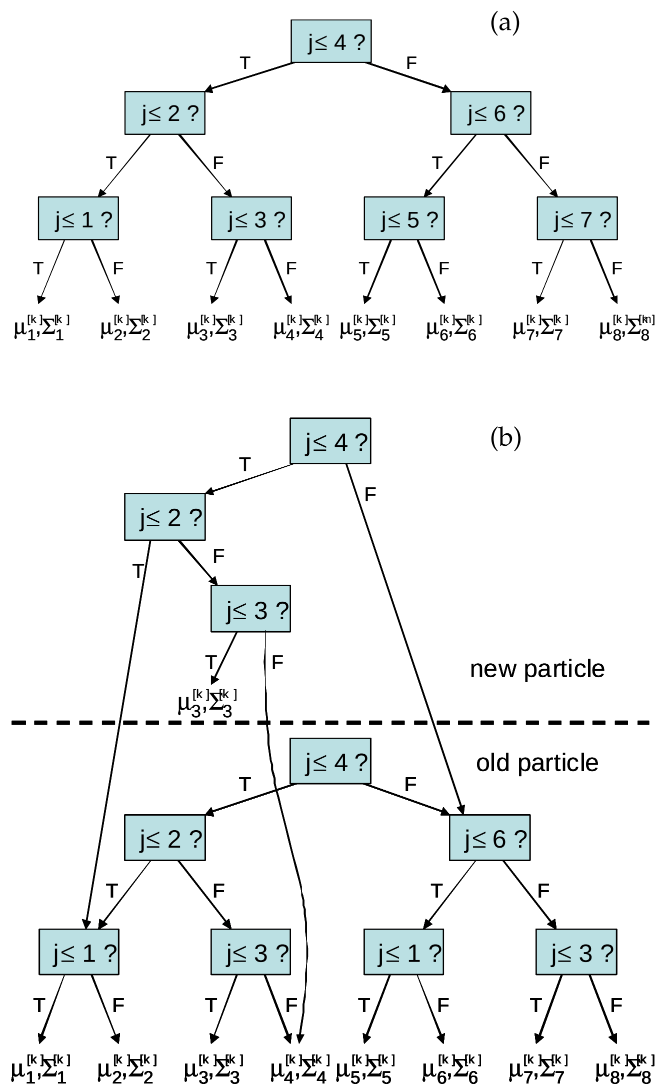


Figure 13.8 (a) A tree representing $N = 8$ feature estimates within a single particle. (b) Generating a new particle from an old one, while modifying only a single Gaussian. The new particle receives only a partial tree, consisting of a path to the modified Gaussian. All other pointers are copied from the generating tree. This can be done in time logarithmic in N.

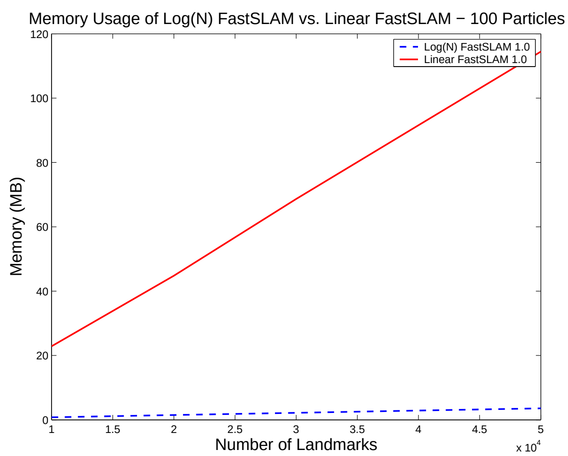


Figure 13.9 Memory requirements for linear and $\log(N)$ version of FastSLAM 1.0.

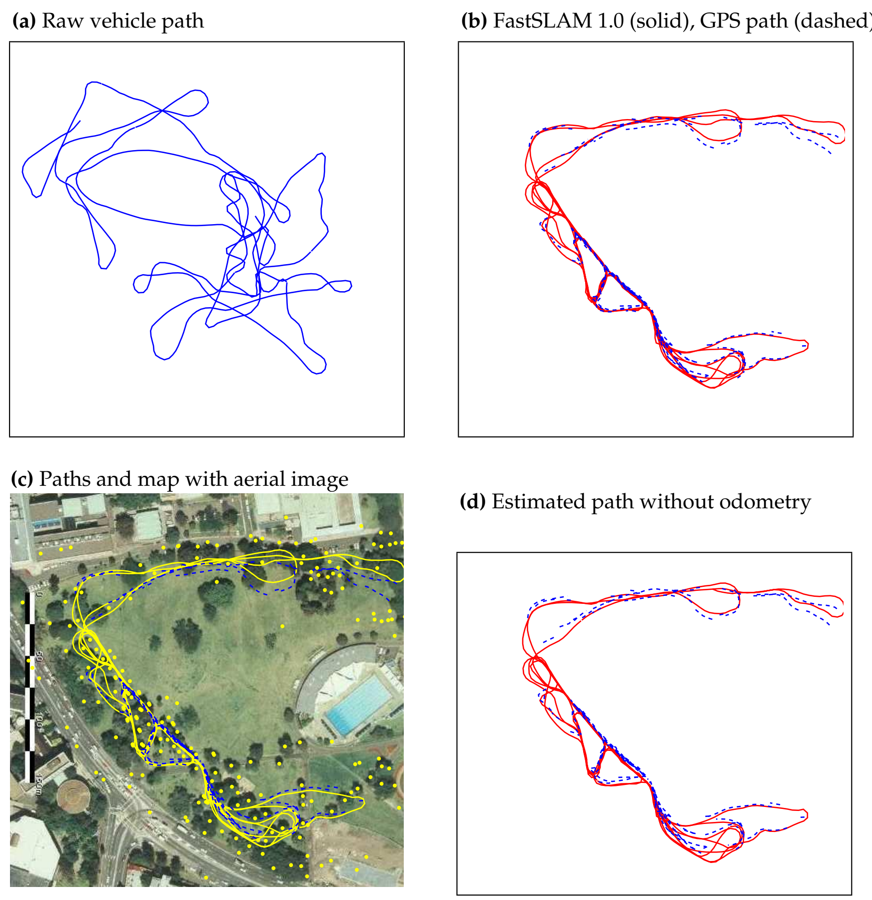


Figure 13.10 (a) Vehicle path predicted by the odometry; (b) True path (dashed line) and FastSLAM 1.0 path (solid line); (c) Victoria Park results overlayed on aerial imagery with the GPS path in blue (dashed), average FastSLAM 1.0 path in yellow (solid), and estimated features as yellow circles. (d) Victoria Park Map created without odometry information. Data and aerial image courtesy of José Guivant and Eduardo Nebot, Australian Centre for Field Robotics.

(a) Map without feature elimination **(b)** Map with feature elimination

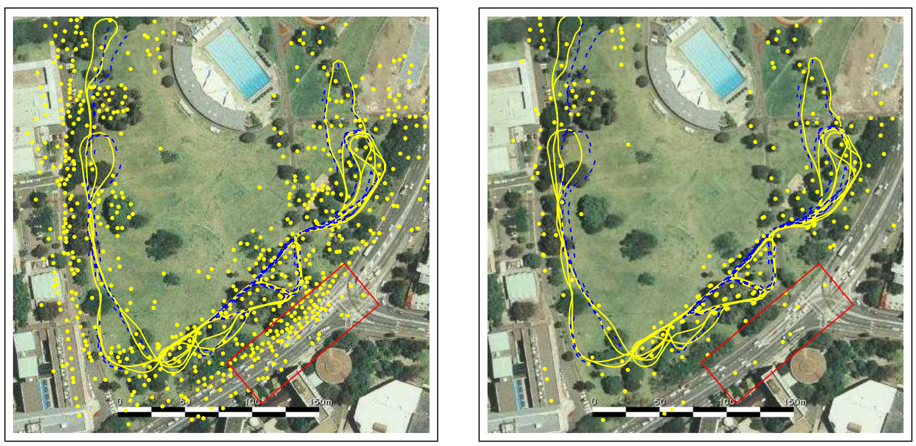


Figure 13.11 FastSLAM 1.0 (a) without and (b) with feature elimination based on negative information.

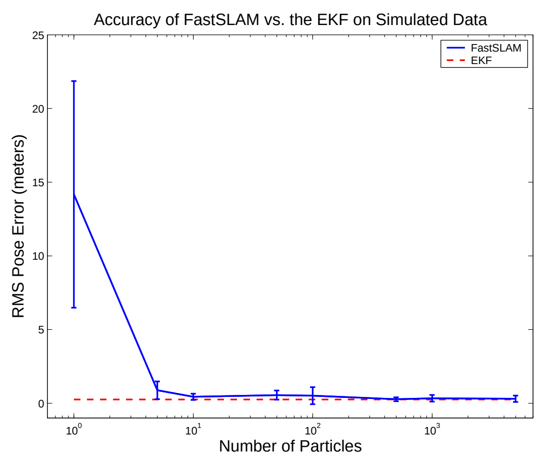


Figure 13.12 A comparison of the accuracy of FastSLAM 1.0 and the EKF on simulated data.

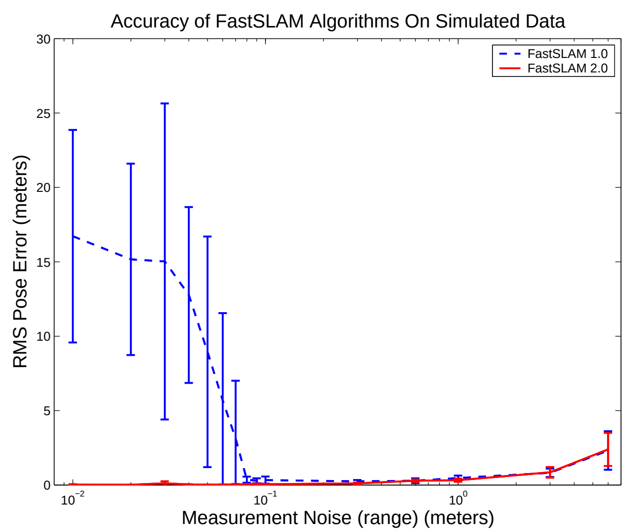


Figure 13.13 FastSLAM 1.0 and 2.0 with varying levels of measurement noise: As to be expected, FastSLAM 2.0 is uniformly superior to FastSLAM 1.0. The difference is particularly obvious for small particle sets, where the improved proposal distribution focuses the particles much better.

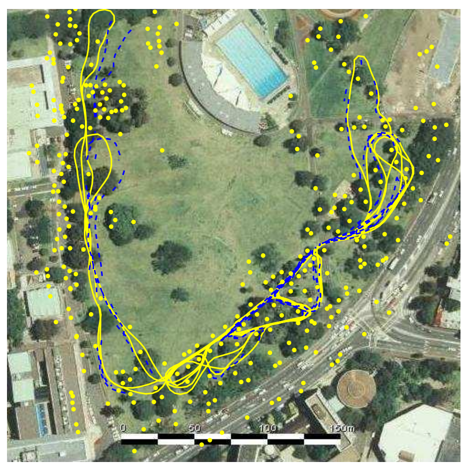


Figure 13.14 Map of Victoria Park by FastSLAM 2.0 with $M = 1$ particle.

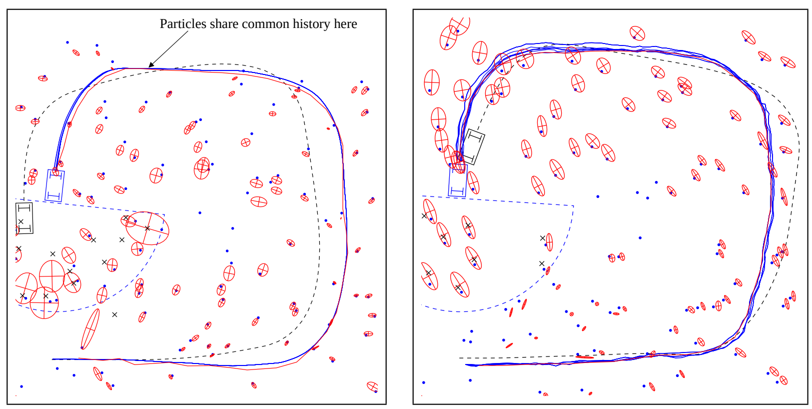


Figure 13.15 FastSLAM 2.0 can close larger loops than FastSLAM 1.0 given a constant number of particles.

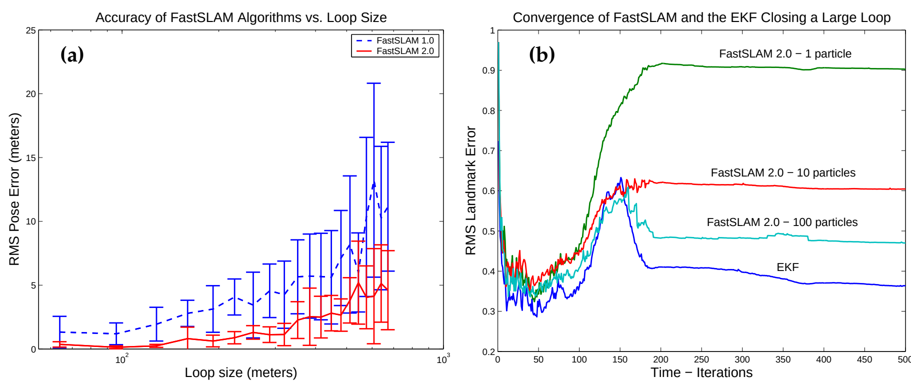


Figure 13.16 (a) Accuracy as a function of loop size: FastSLAM 2.0 can close larger loops than FastSLAM 1.0 given a fixed number of particles. (b) Comparison of the convergence speed of FastSLAM 2.0 and the EKF.

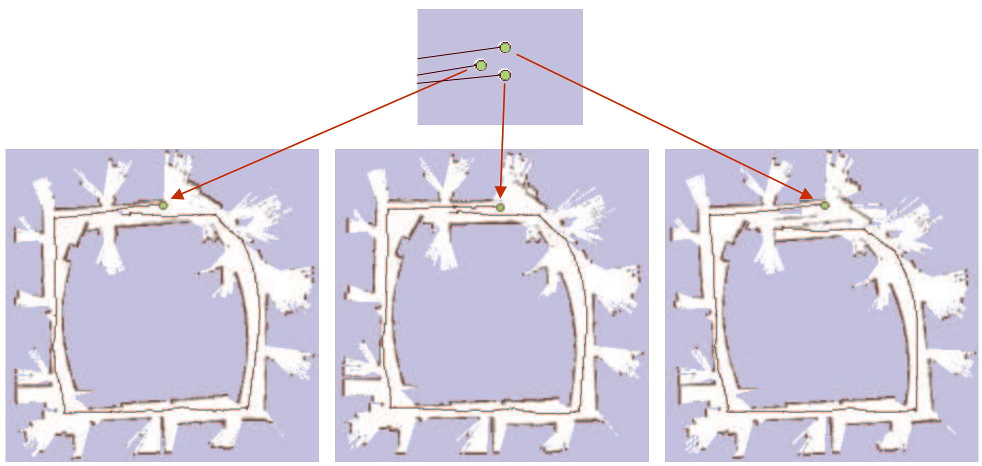


Figure 13.17 Application of the grid-based variant of the FastSLAM algorithm. Each particle carries its own map and the importance weights of the particles are computed based on the likelihood of the measurements given the particle's own map.

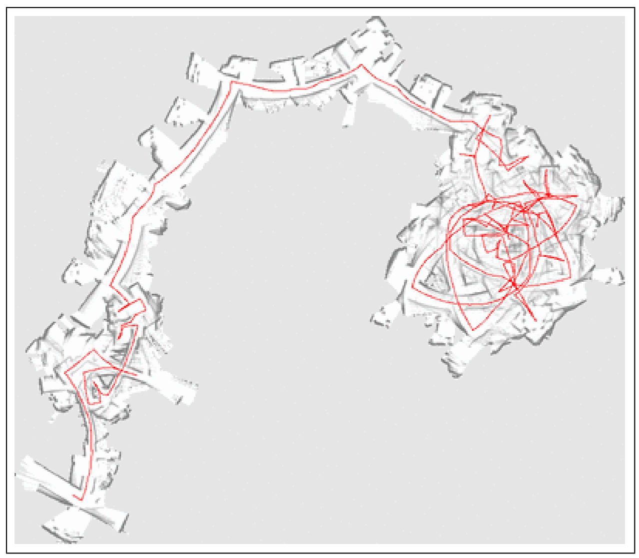


Figure 13.18 Occupancy grid map generated from laser range data and based on pure odometry. All images courtesy of Dirk Hähnel, University of Freiburg.

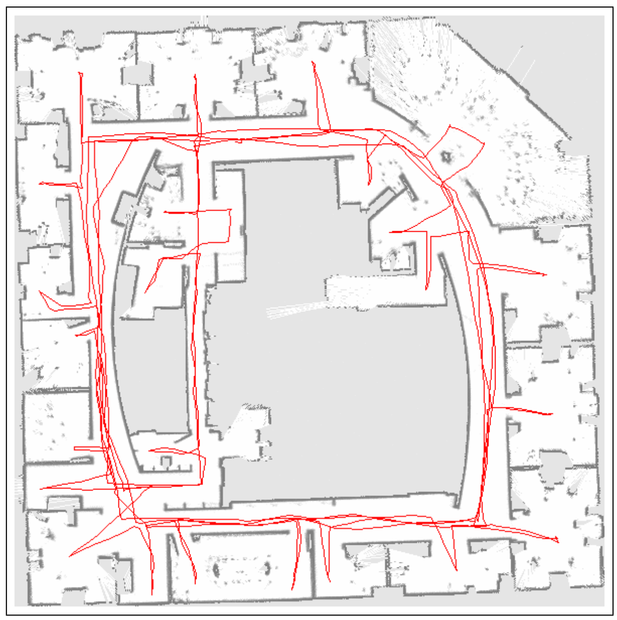


Figure 13.19 Occupancy grid map corresponding to the particle with the highest accumulated importance weight obtained by the algorithm listed in Table **??** from the data depicted in Figure 13.18. The number of particles to create this experiment was 500. Also depicted in this image is the path represented by the particle with the maximum accumulated importance weight.

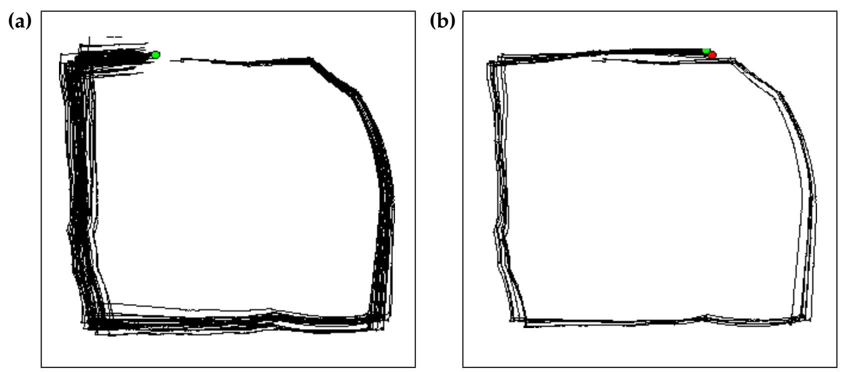


Figure 13.20 Trajectories of all samples shortly before (left) and after (right) closing the outer loop of the environment depicted in Figure 13.19. Images courtesy of Dirk Hähnel, University of Freiburg.

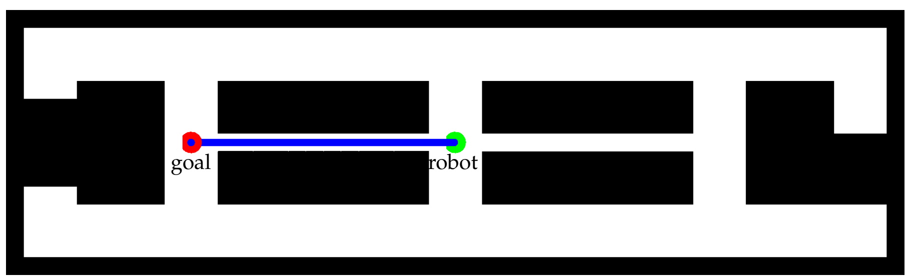


Figure 14.1 Near-symmetric environment with narrow and wide corridors. The robot starts at the center with unknown orientation. Its task is to move to the goal location on the left.

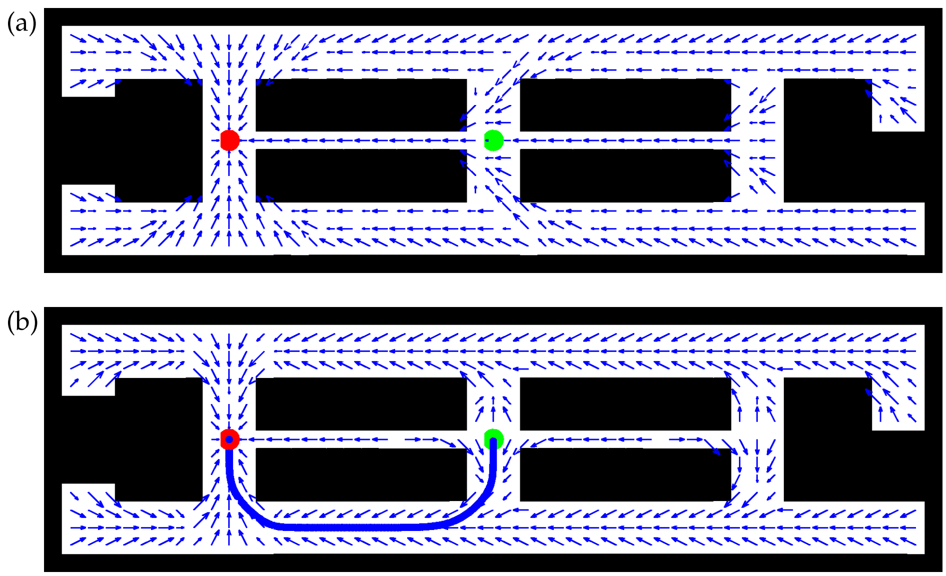


Figure 14.2 The value function and control policy for an MDP with (a) deterministic and (b) nondeterministic action effects. Under the deterministic model, the robot is perfectly fine to navigate through the narrow path; it prefers the longer path when action outcomes are uncertain, to reduce the risk of colliding with a wall. Panel (b) also shows a path.

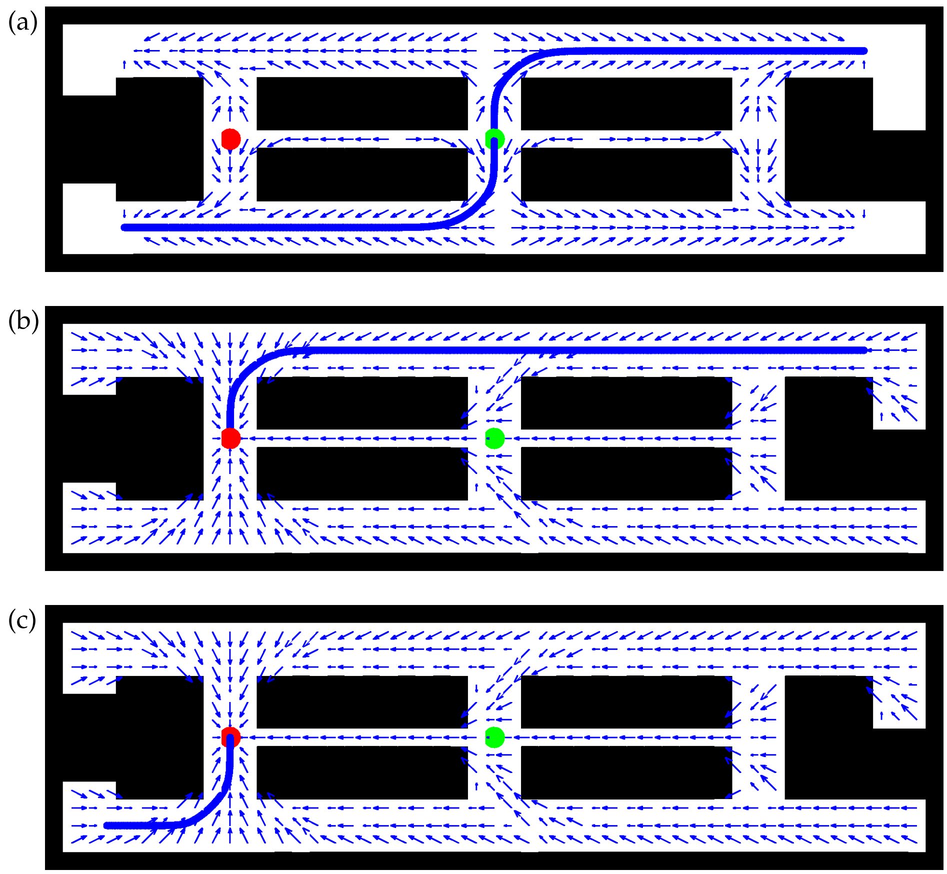


Figure 14.3 Knowledge gathering actions in POMDPs: To reach its goal with more than 50% chance, the belief space planner first navigates to a location where the global orientation can be determined. Panel (a) shows the corresponding policy, and a possible path the robot may take. Based on its location, the robot will then find itself in panel (b) or (c), from where it can safely navigate to the goal.

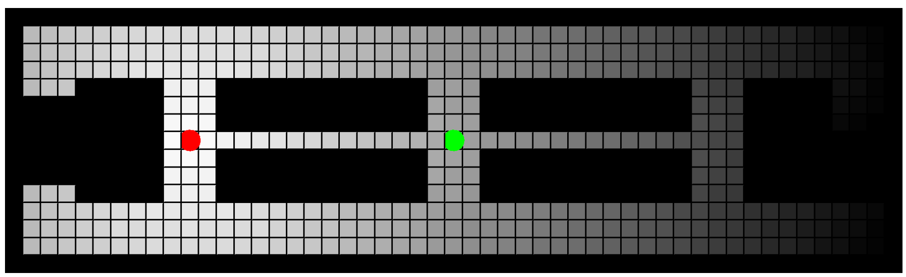


Figure 14.4 An example of an infinite-horizon value function T_∞, assuming that the goal state is an "absorbing state." This value function induces the policy shown in Figure 14.2a.

(a) (b)

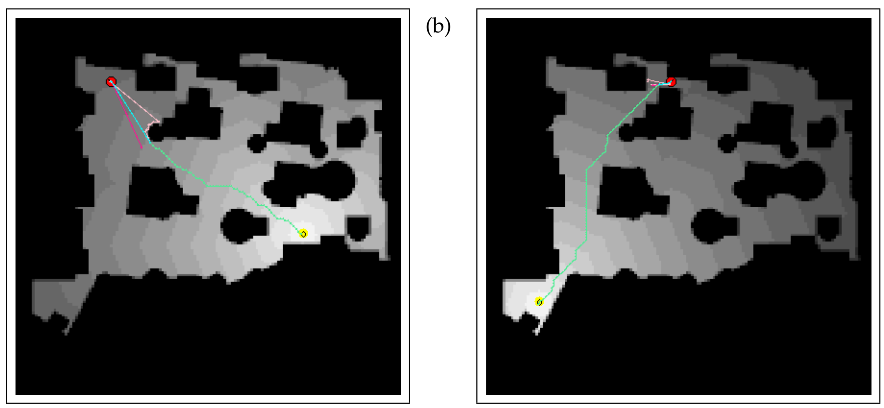


Figure 14.5 Example of value iteration over state spaces in robot motion. Obstacles are shown in black. The value function is indicated by the gray shaded area. Greedy action selection with respect to the value function lead to optimal control, assuming that the robot's pose is observable. Also shown in the diagrams are example paths obtained by following the greedy policy.

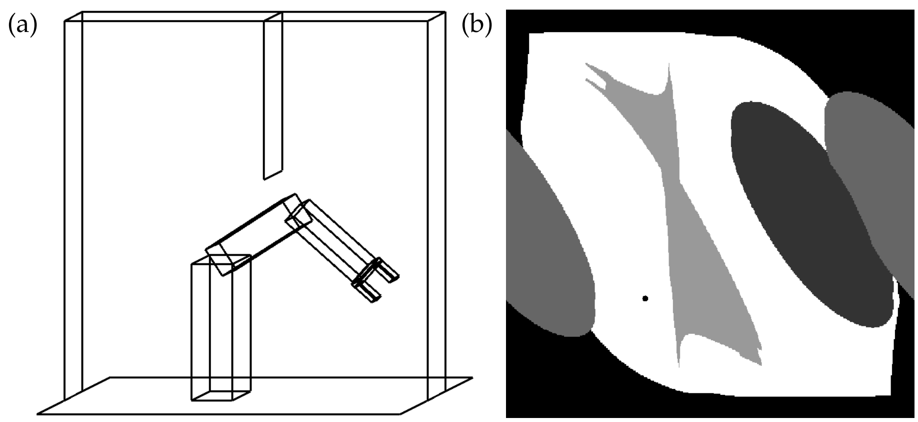


Figure 14.6 (a) 2-DOF robot arm in an environment with obstacles. (b) The *configuration space* of this arm: the horizontal axis corresponds to the shoulder joint, and the vertical axis to its elbow joint configuration. Obstacles are shown in gray. The small dot in this diagram corresponds to the configuration on the left.

(a) (b)

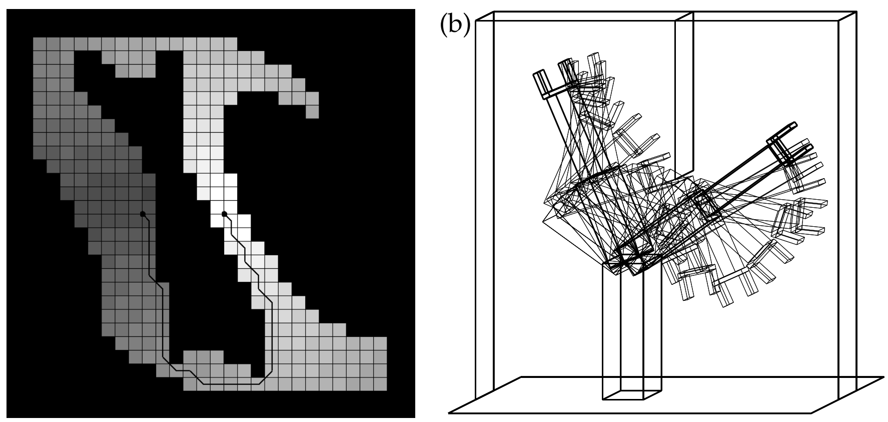


Figure 14.7 (a) Value iteration applied to a coarse discretization of the configuration space. (b) Path in workspace coordinates. The robot indeed avoids the vertical obstacle.

(a) (b)

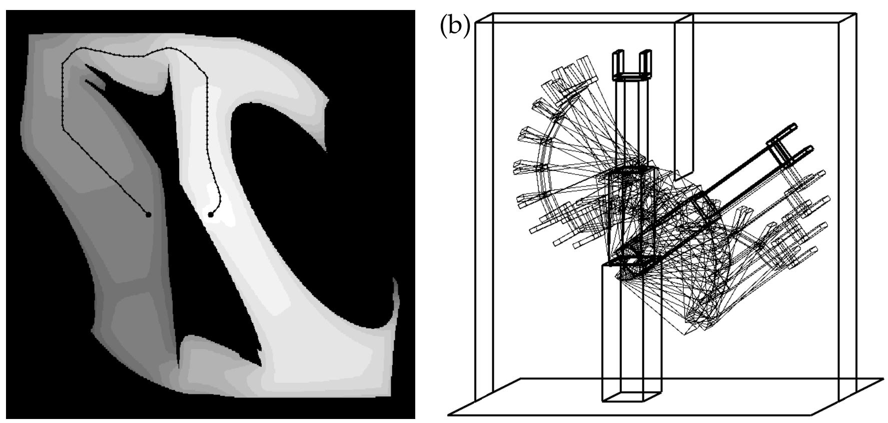


Figure 14.8 (a) Probabilistic value iteration, here over a fine-grained grid. (b) The corresponding path.

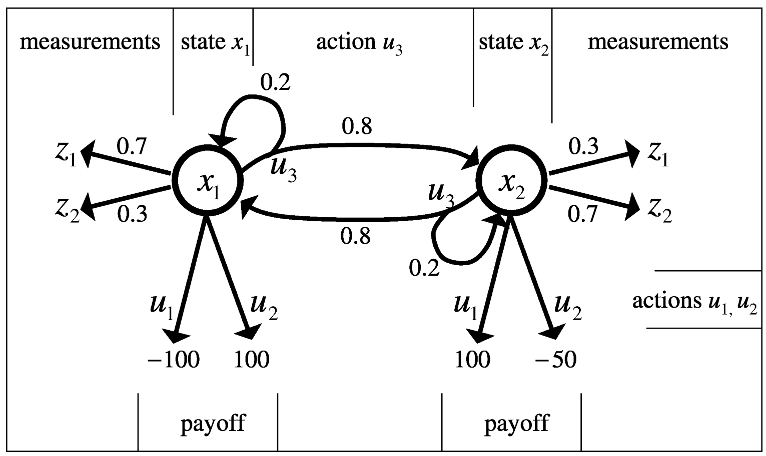


Figure 15.1 The two-state environment used to illustrate value iteration in belief space.

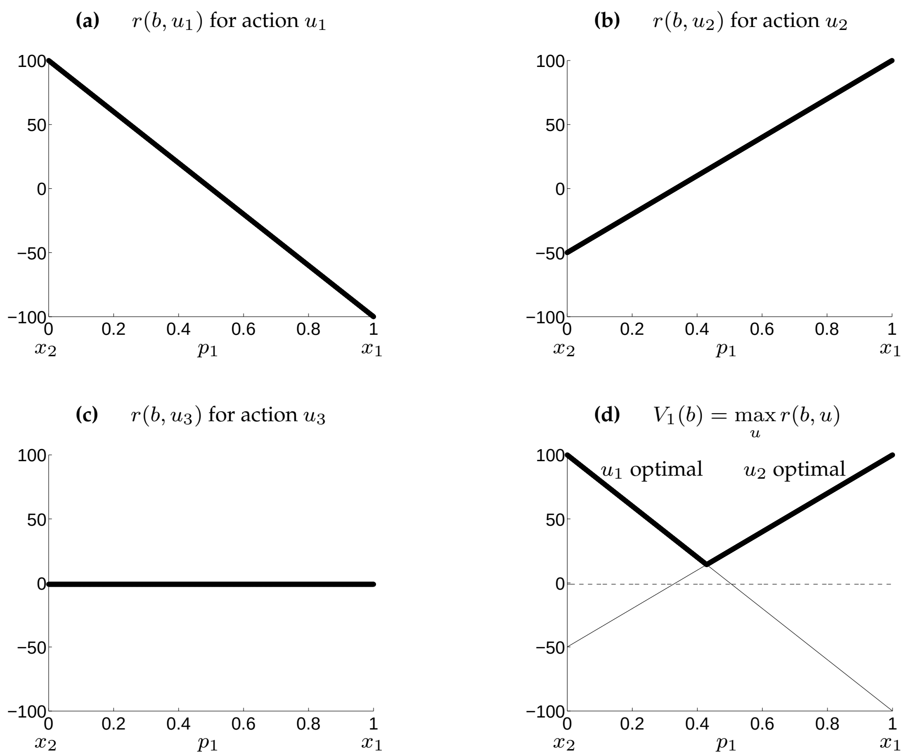


Figure 15.2 Diagrams (a), (b), and (c) depict the expected payoff r as a function of the belief state parameter $p_1 = b(x_1)$, for each of the three actions u_1, u_2, and u_3. (d) The value function at horizon $T = 1$ corresponds to the maximum of these three linear functions.

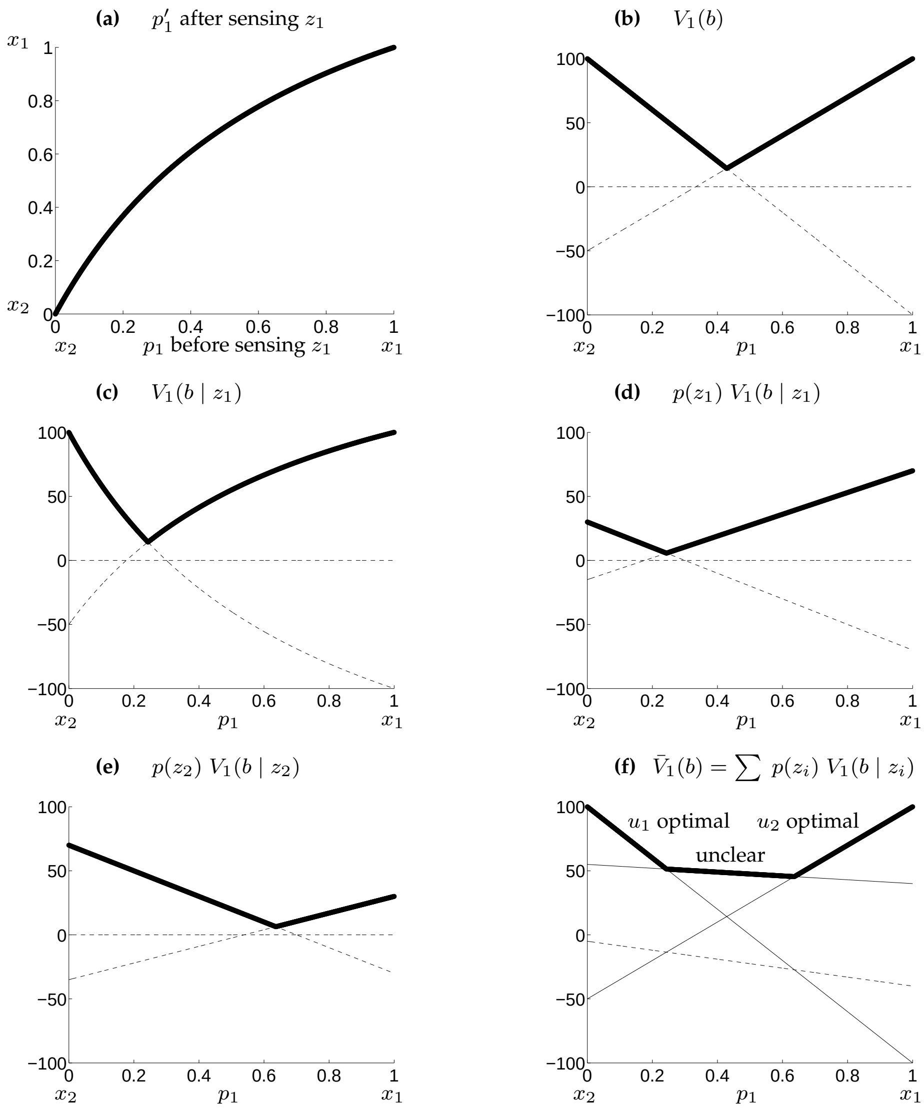


Figure 15.3 The effect of sensing on the value function: (a) The belief after sensing z_1 as a function of the belief before sensing z_1. Sensing z_1 makes the robot more confident that the state is x_1. Projecting the value function in (b) through this nonlinear function results in the non-linear value function in (c). (d) Dividing this value function by the probability of observing z_1 results in a piecewise linear function. (e) The same piecewise linear function for measurement z_2. (f) The expected value function after sensing.

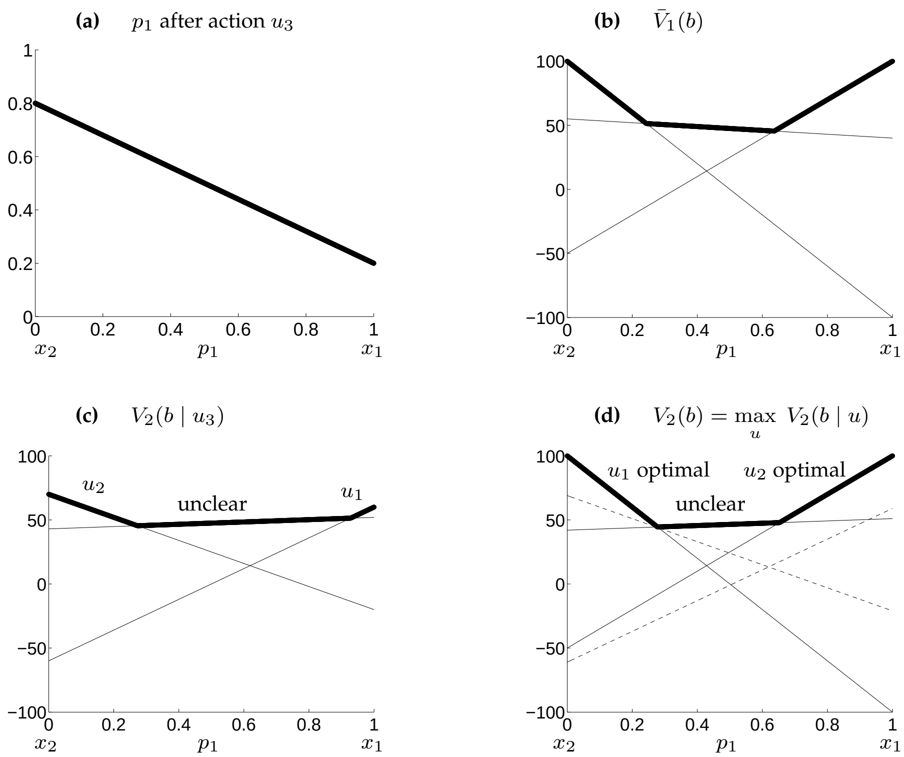


Figure 15.4 (a) The belief state parameter p_1' after executing action u_3, as a function of the parameter p_1 before the action. Propagating the belief shown in (b) through the inverse of this mapping results in the belief shown in (c). (d) The value function V_2 obtained by maximizing the propagated belief function, and the payoff of the two remaining actions, u_1 and u_2.

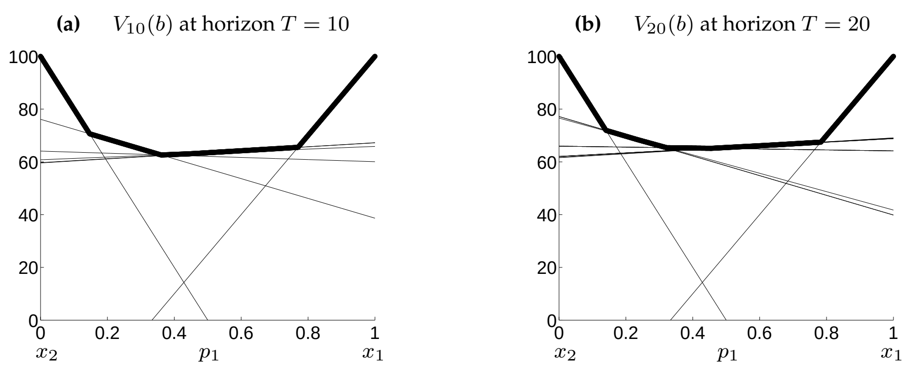


Figure 15.5 The value function V for horizons $T = 10$ and $T = 20$. Note that the vertical axis in these plots differs in scale from previous depictions of value functions.

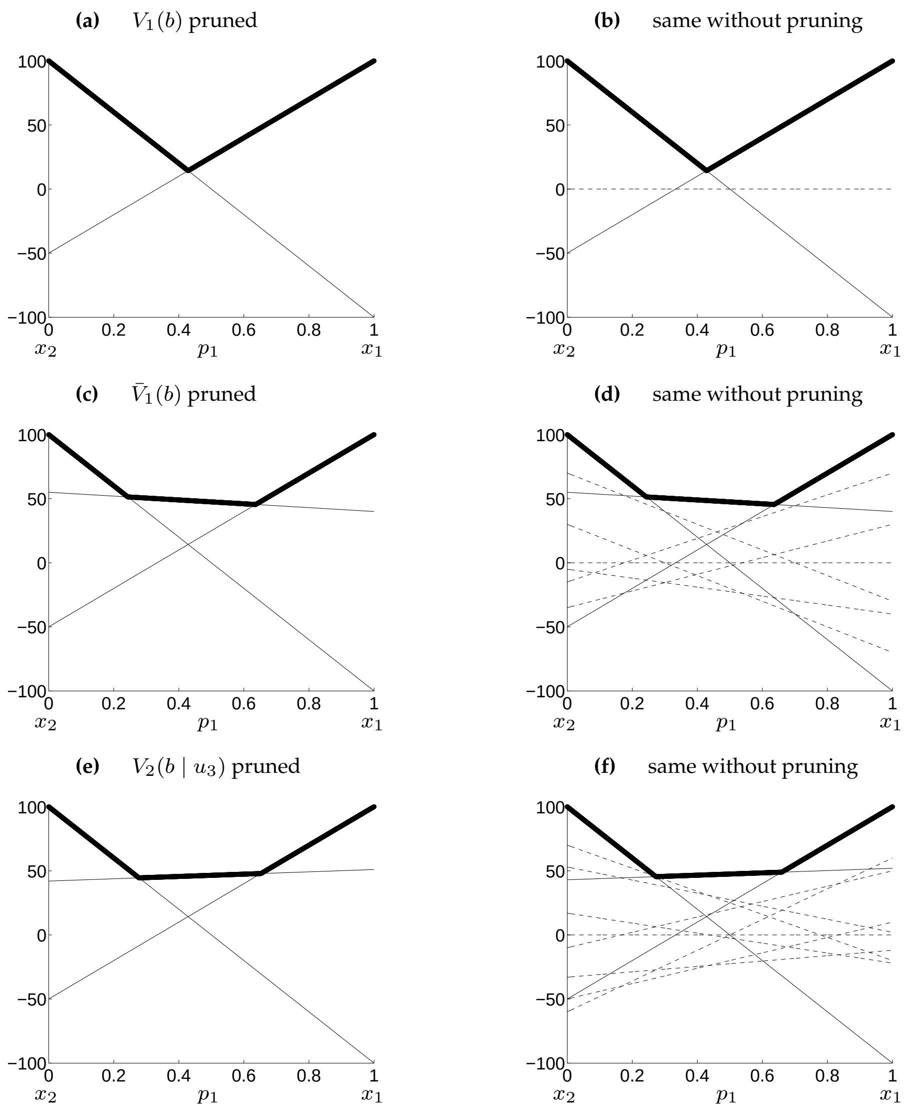


Figure 15.6 Comparison of an exact pruning algorithm (left row) versus a non-pruning POMDP algorithm (right row), for the first few steps of the POMDP planning algorithm. Obviously, the number of linear constraints increases dramatically without pruning. At $T = 20$, the unpruned value function is defined over $10^{547,864}$ linear functions, whereas the pruned one only uses 13 such functions.

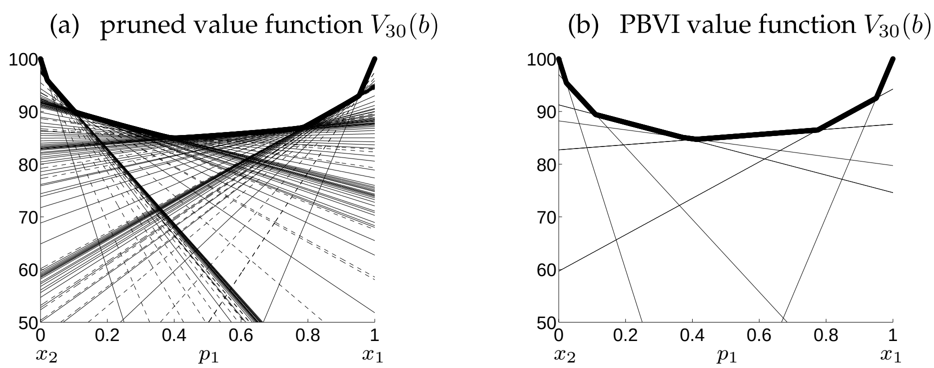


Figure 15.7 The benefit of point-based value iteration over general value iteration: Shown in (a) is the exact value function at horizon $T = 30$ for a different example, which consists of 120 constraints, after pruning. On the right is the result of the PBVI algorithm retaining only 11 linear functions. Both functions yield virtually indistinguishable results when applied to control.

(a)

(b)

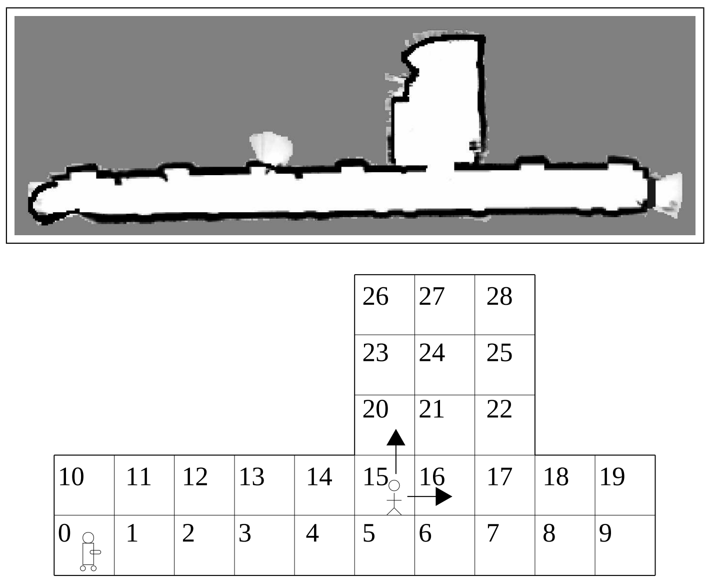


Figure 15.8 Indoor environment, in which we seek a control policy for finding a moving intruder. (a) Occupancy grid map, and (b) discrete state set used by the POMDP. The robot tracks its own pose sufficiently well that the pose uncertainty can be ignored. The remaining uncertainty pertains to the location of the person. Courtesy of Joelle Pineau, McGill University.

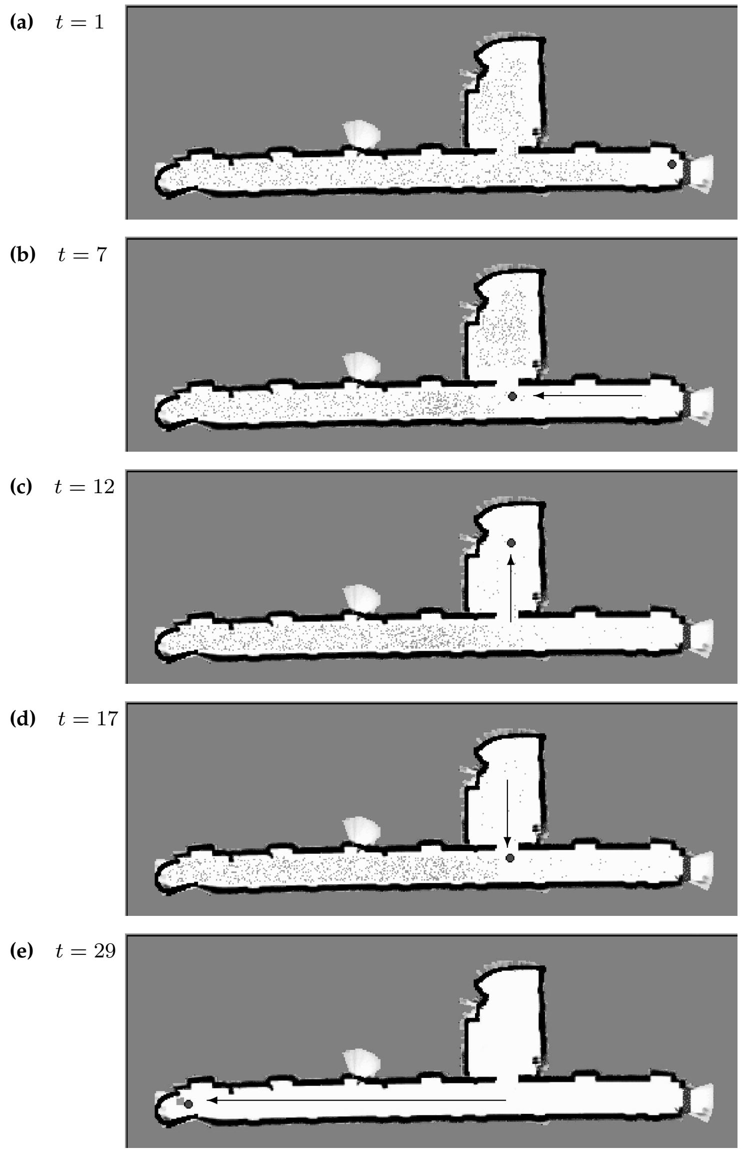


Figure 15.9 A successful search policy. Here the tracking of the intruder is implemented via a particle filter, which is then projected into a histogram representation suitable for the POMDP. The robot first clears the room on the top, then proceeds down the hallway. Courtesy of Joelle Pineau, McGill University.

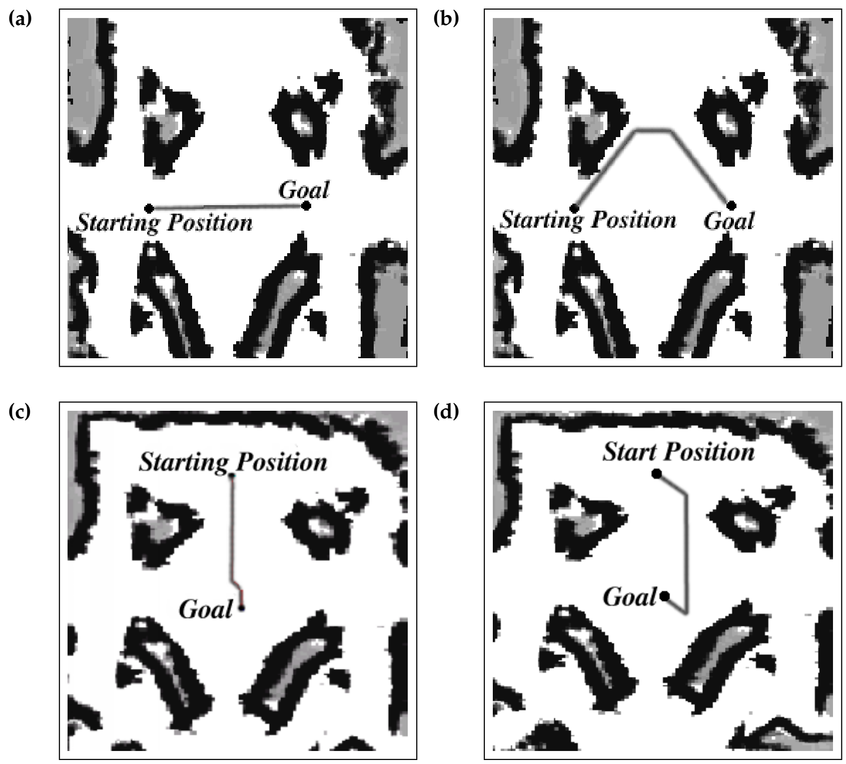


Figure 16.1 Examples of robot paths in a large, open environment, for two different configurations (top row and bottom row). The diagrams (a) and (c) show paths generated by a conventional dynamic programming path planner that ignores the robot's perceptual uncertainty. The diagrams (b) and (d) are obtained using the augmented MDP planner, which anticipates uncertainty and avoids regions where the robot is more likely to get lost. Courtesy of Nicholas Roy, MIT.

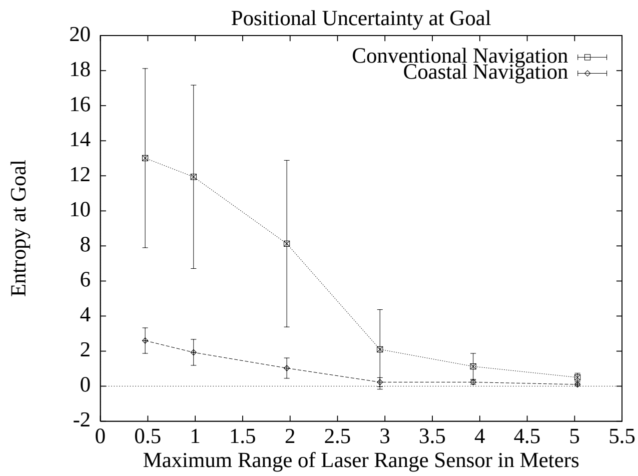


Figure 16.2 Performance comparison of MDP planning and Augmented MDP planning. Shown here is the uncertainty (entropy) at the goal location as a function of the sensor range. Courtesy of Nicholas Roy, MIT.

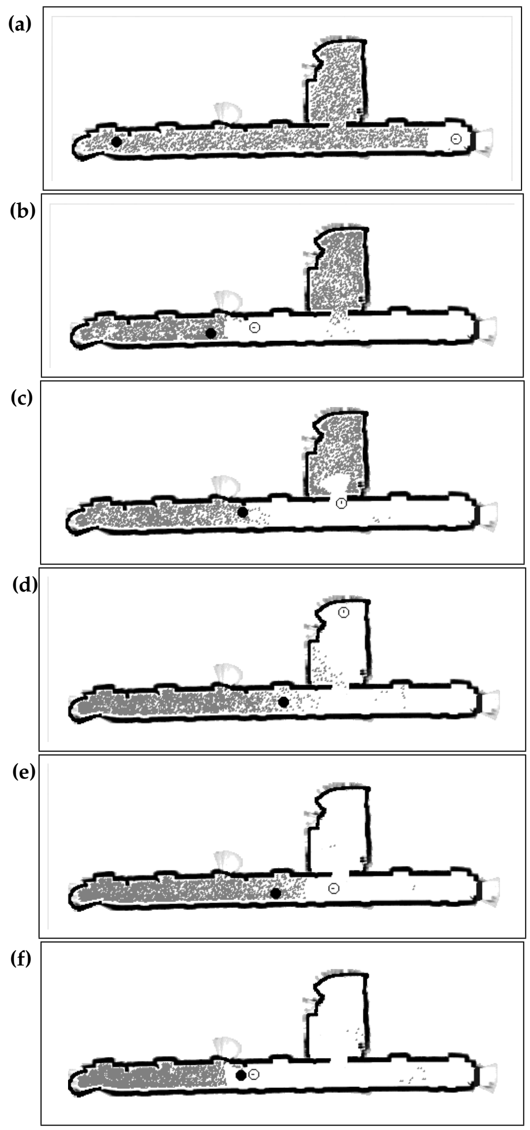


Figure 16.3 The policy computed using an advanced version of AMDP, with a learned state representation. The task is to find an intruder. The gray particles are drawn from the distribution of where the person might be, initially uniformly distributed in (a). The black dot is the true (unobservable) position of the person. The open circle is the observable position of the robot. This policy succeeds with high likelihood. Courtesy of Nicholas Roy, MIT, and Geoffrey Gordon, CMU.

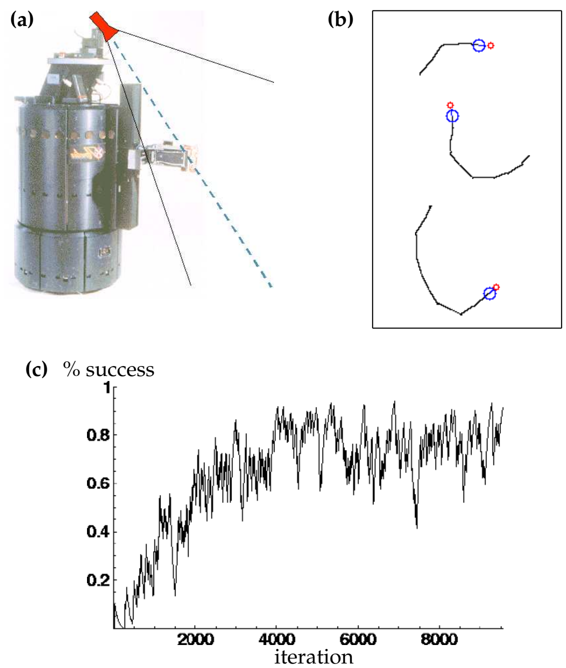


Figure 16.4 A robotic find and fetch task: (a) The mobile robot with gripper and camera, holding the target object. (b) 2-D trajectory of three successful policy executions, in which the robot rotates until it sees the object, and then initiates a successful grasp action (c) success rate as a function of number of planning steps, evaluated in simulation.

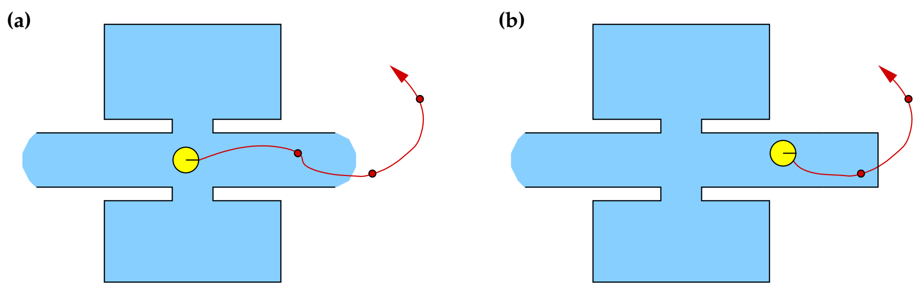


Figure 17.1 Unpredictability of the exploration problem: A robot in (a) might anticipate a sequence of three controls, but whether or not this sequence is executable depends on the things the robot finds out along the way. Any exploration policy has to be highly reactive.

(a) Environment with an example posterior. **(b)** Effect of exploration action.

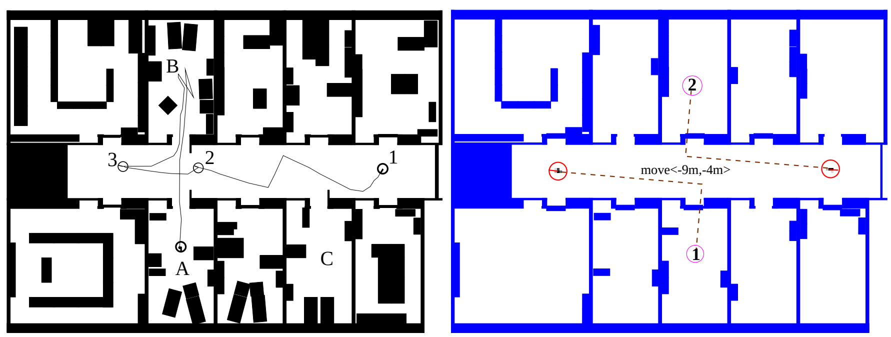
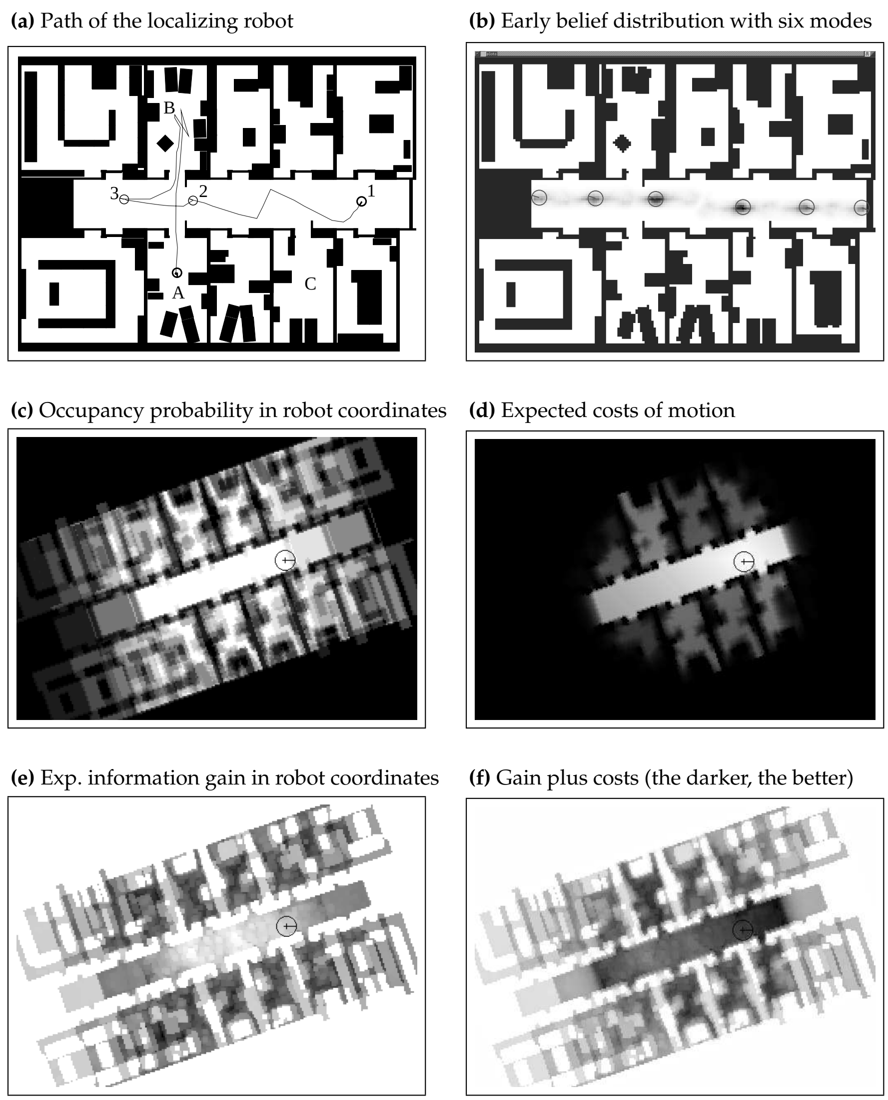


Figure 17.2 (a) Active localization in a symmetric environment: Shown here is an environment with a symmetric corridor, but an asymmetric arrangement of rooms, labeled A, B, and C. This figure also shows an exploration path. (b) An example of the exploration action "go backward 9 meters, go left 4 meters." If the robot's pose posterior possesses two distinct modes as shown here, the actual control in global world coordinates might lead to two different places.



(a) Path of the localizing robot

(b) Early belief distribution with six modes

(c) Occupancy probability in robot coordinates

(d) Expected costs of motion

(e) Exp. information gain in robot coordinates

(f) Gain plus costs (the darker, the better)

Figure 17.3 Illustration of active localization. This figure displays a number of auxiliary functions for computing the optimal action, for a multi-hypothesis pose distribution.

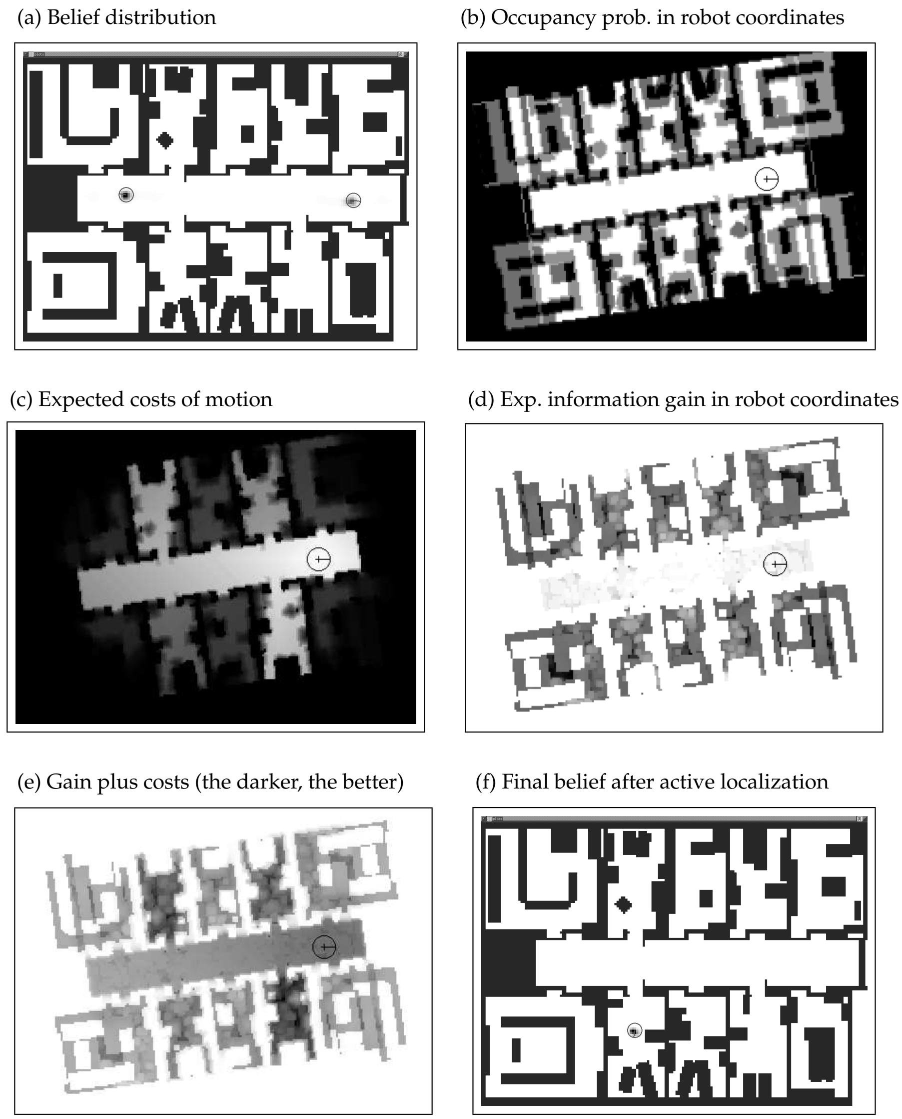


Figure 17.4 Illustration of active localization at some later point in time, for a belief with two distinct modes.

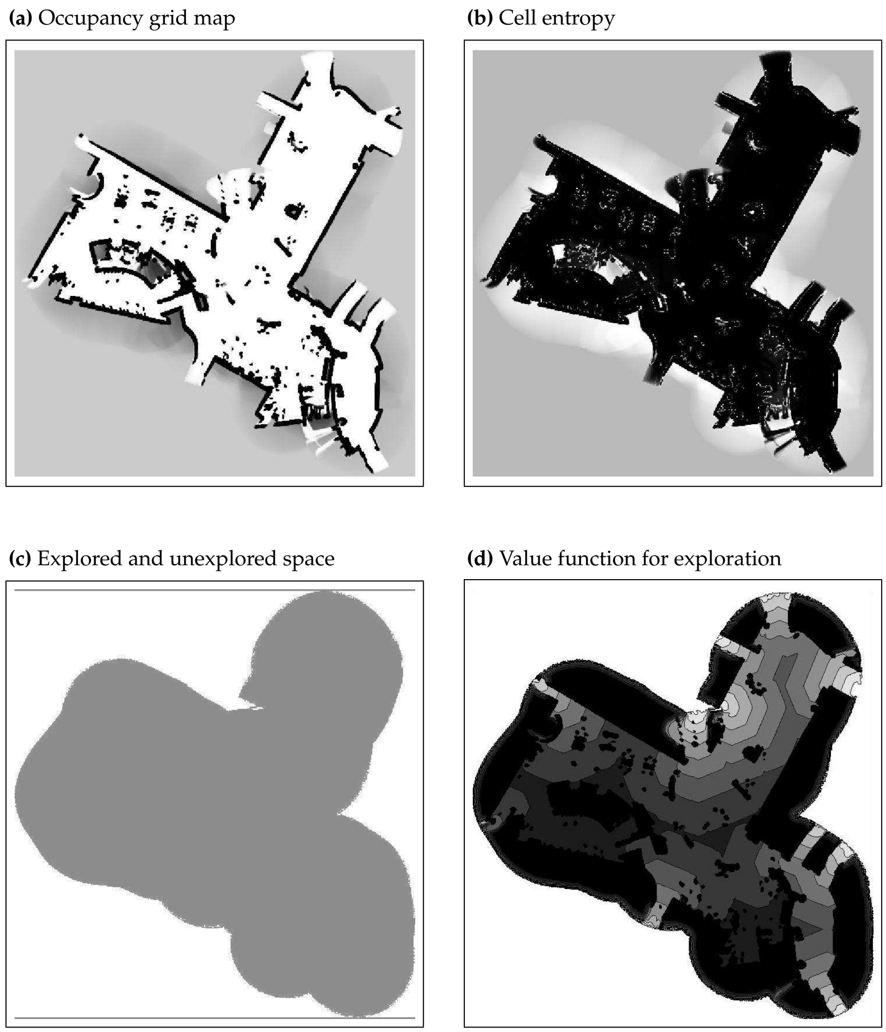


Figure 17.5 Example of the essential step in exploration for mapping. (a) shows a partial grid map; (b) depicts the map entropy; (c) shows the space for which we have zero information; and (d) displays the value function for optimal exploration.

(a) Map segment **(b)** Entropy **(c)** Exp. information gain

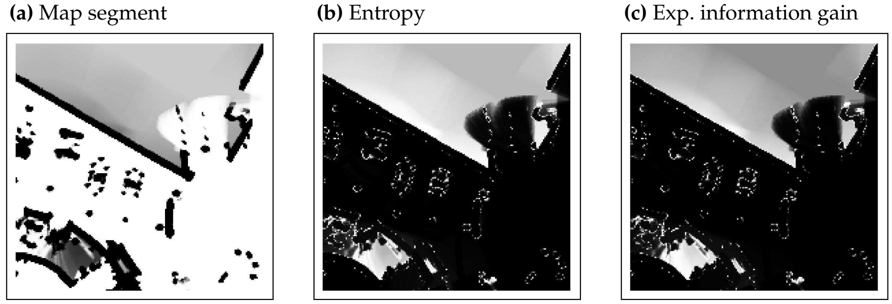
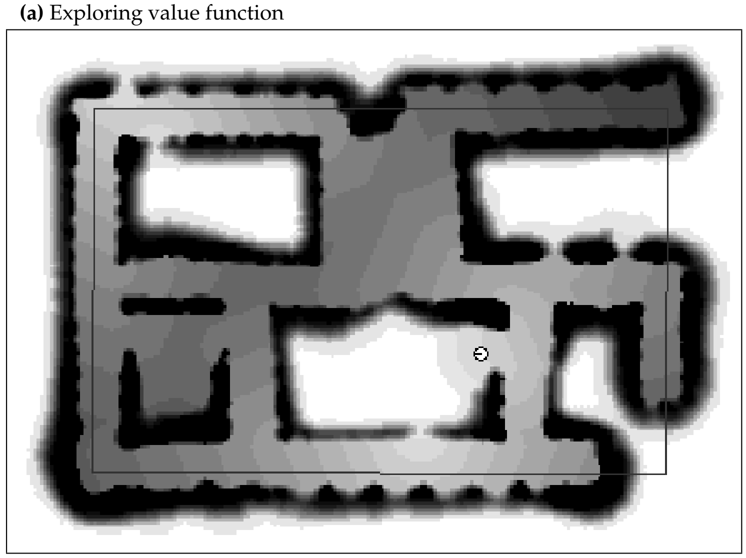


Figure 17.6 Map, entropy and expected information gain. This figure illustrates that with the appropriate scaling, entropy and expected information gain are nearly indistinguishable.

(a) Exploring value function



(b) Exploration path

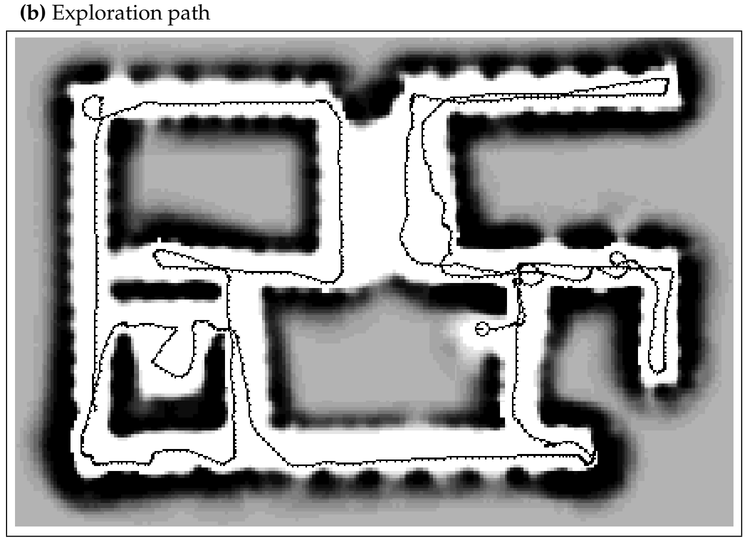


Figure 17.7 Illustration of autonomous exploration. (a) Exploration values V, computed by value iteration. White regions are completely unexplored. By following the gray-scale gradient, the robot moves to the next unexplored area on a minimum-cost path. The large black rectangle indicates the global wall orientation θ_{wall}. (b) Actual path traveled during autonomous exploration, along with the resulting metric map.

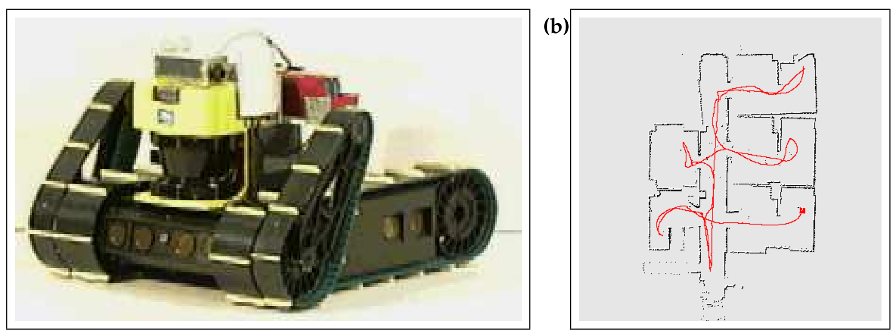
(a)  (b)

Figure 17.8 (a) Urban robot for indoor and outdoor exploration. The urban robot's odometry happens to be poor. (b) Exploration path of the autonomously exploring robot, using the exploration techniques described in the text.

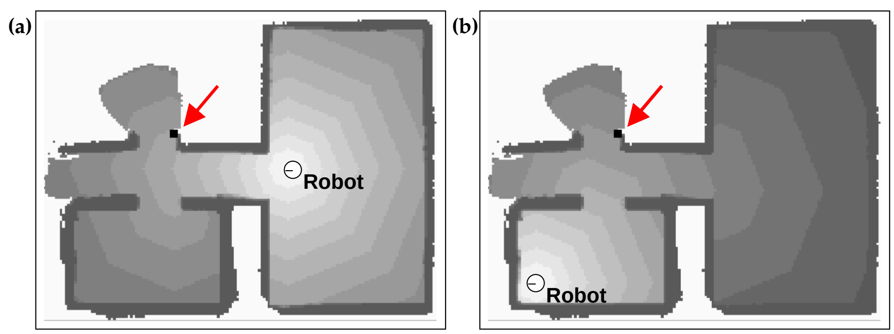


Figure 17.9 Two robots exploring an environment. Without any coordination both vehicles would decide to approach the same target location. Each image shows the robot, the map, and its value function. The black rectangle indicates the target points with minimum cost.

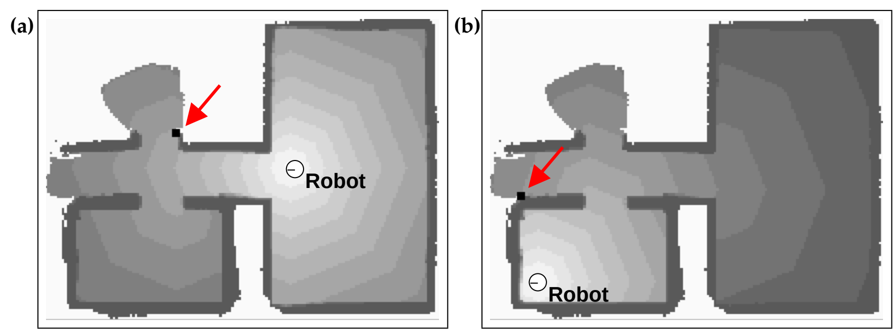


Figure 17.10 Target positions obtained using the coordination approach. In this case the target point for the second robot is to the left in the corridor.

Figure 17.11 Coordinated exploration by a team of mobile robots. The robots distribute themselves throughout the environment.

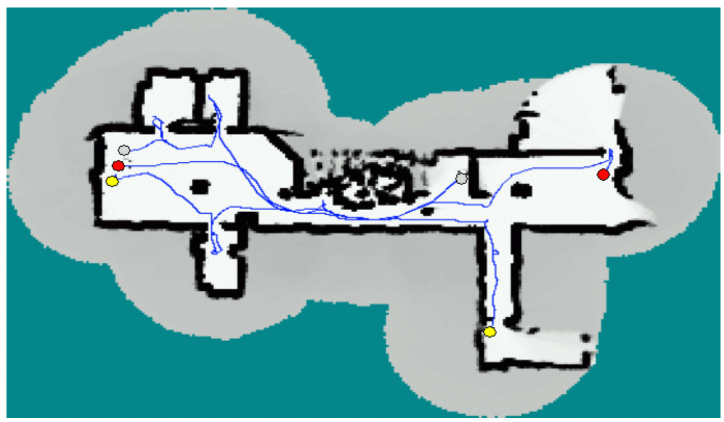


Figure 17.12 Map of a $62 \times 43m^2$ large environment learned by three robots in 8 minutes.

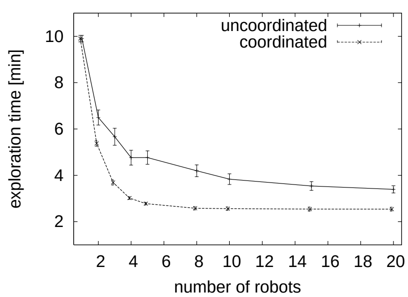


Figure 17.13 Exploration time obtained in a simulation experiment in which robot teams of different sizes explore the environment shown in the left image.

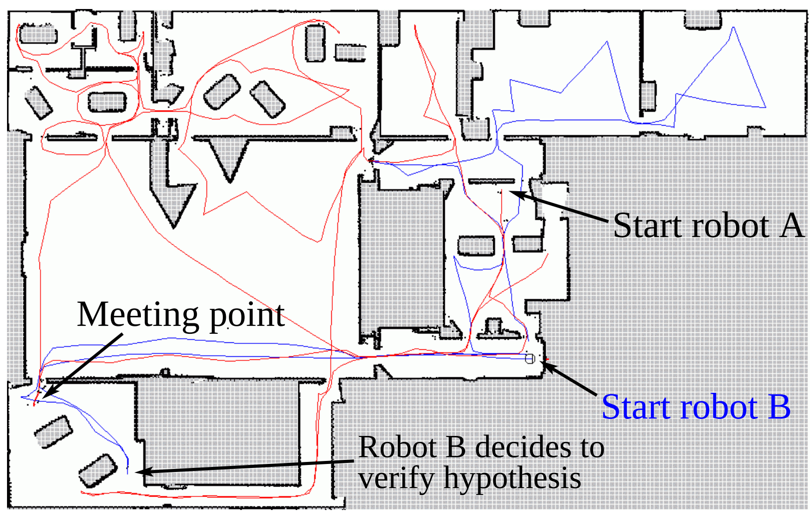


Figure 17.14 Coordinated exploration from unknown start locations. The robots establish a common frame of reference by estimating and verifying their relative locations using a rendezvous approach. Once they meet, they share a map and coordinate their exploration. Courtesy of Jonathan Ko and Benson Limketkai.

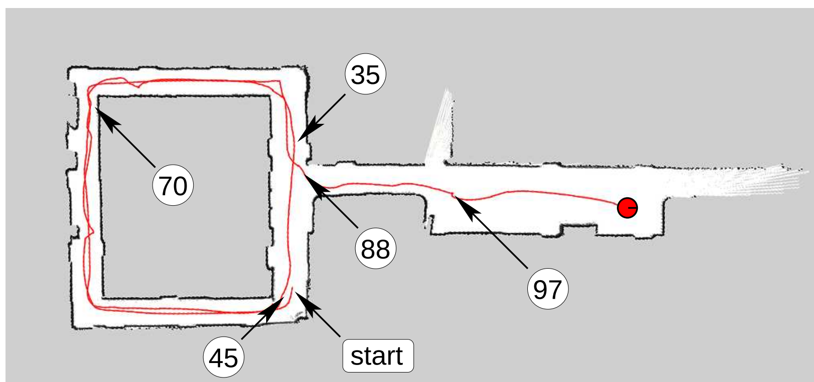


Figure 17.15 A mobile robot explores an environment with a loop. The robot starts in the lower right corner of the loop. After traversing it, it decides to follow the previous trajectory again in order to reduce its uncertainty. Then it continues to explore the corridor. Courtesy of Cyrill Stachniss, University of Freiburg.

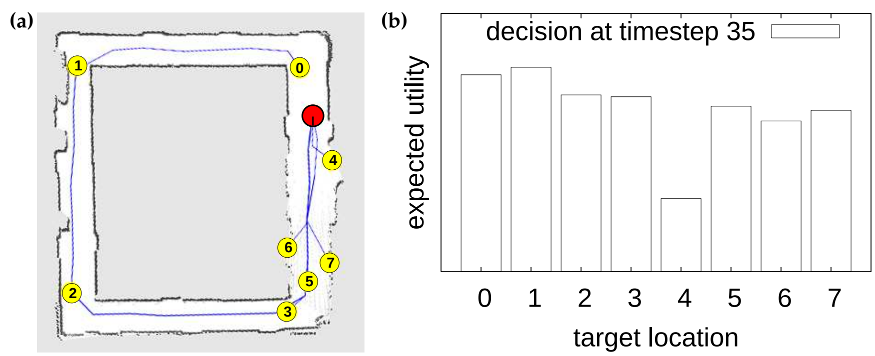


Figure 17.16 In this situation the robot determines the expected utility of possible actions. (a) the exploration actions considered by the robot; (b) the expected utility of each action. Action 1 is selected because it maximizes the expected utility. Courtesy of Cyrill Stachniss, University of Freiburg.

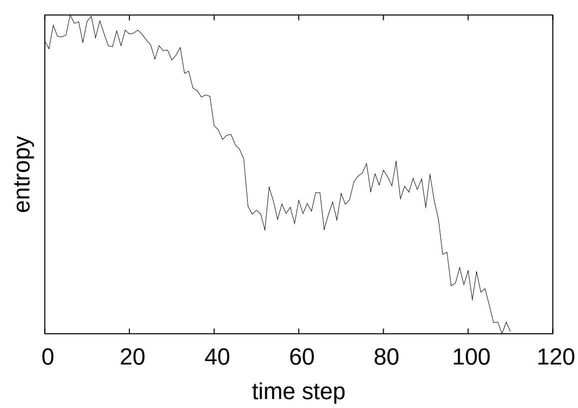


Figure 17.17 The evolution of the entropy during the exploration experiment shown in Figure 17.15. Courtesy of Cyrill Stachniss, University of Freiburg.