Accepted for Publication in *Artificial Intelligence*

Learning Maps for Indoor Mobile Robot Navigation ⋆

Sebastian Thrun

Computer Science Department and Robotics Institute
Carnegie Mellon University, Pittsburgh

Abstract

Autonomous robots must be able to learn and maintain models of their environments. Research on mobile robot navigation has produced two major paradigms for mapping indoor environments: grid-based and topological. While grid-based methods produce accurate metric maps, their complexity often prohibits efficient planning and problem solving in large-scale indoor environments. Topological maps, on the other hand, can be used much more efficiently, yet accurate and consistent topological maps are often difficult to learn and maintain in large-scale environments, particularly if momentary sensor data is highly ambiguous. This paper describes an approach that integrates both paradigms: grid-based and topological. Grid-based maps are learned using artificial neural networks and naive Bayesian integration. Topological maps are generated on top of the grid-based maps, by partitioning the latter into coherent regions. By combining both paradigms, the approach presented here gains advantages from both worlds: accuracy/consistency and efficiency. The paper gives results for autonomous exploration, mapping and operation of a mobile robot in populated multi-room environments.

⋆ This research was sponsored in part by the National Science Foundation under award IRI-9313367, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Darpa Advanced Research Projects Agency (DARPA) under grant number F33615-93-1-1330. We also acknowledge financial support by Daimler Benz Corp.

1 Introduction

To efficiently carry out complex missions in indoor environments, autonomous mobile robots must be able to acquire and maintain models of their environments. The problem of acquiring models is difficult and far from being solved. The following factors impose practical limitations on a robot's ability to learn and use accurate models:

(i) **Sensors.** Sensors often are not capable of directly measuring the quantity of interest. For example, cameras measure color, brightness and saturation of light, whereas for navigation one might be interested in assertions such as *"there is a door in front of the robot."*
(ii) **Perceptual limitations.** The perceptual range of most sensors (such as ultrasonic transducers, cameras) is limited to a small range around the robot. To acquire global information, the robot has to actively explore its environment.
(iii) **Sensor noise.** Sensor measurements are typically corrupted by noise. Often, the distribution of this noise is not known.
(iv) **Drift/slippage.** Robot motion is inaccurate. Unfortunately, odometric errors accumulate over time. For example, even the smallest rotational errors can have huge effects on subsequent translational errors when estimating the robot's position.
(v) **Complexity and dynamics.** Robot environments are complex and dynamic, making it principally impossible to maintain exact models and to predict accurately.
(vi) **Real-time requirements.** Time requirements often demand that internal models must be simple and easily accessible. For example, accurate fine-grain CAD models of complex indoor environments are often inappropriate if actions have to be generated in real-time.

Recent research has produced two fundamental paradigms for modeling indoor robot environments: the *grid-based (metric) paradigm* and the *topological paradigm*. Grid-based approaches, such as those proposed by Moravec/Elfes [31,32,70] and Borenstein/Koren [8] and many others, represent environments by evenly-spaced grids. Each grid cell may, for example, indicate the presence of an obstacle in the corresponding region of the environment. Topological approaches, such a those proposed by Kuipers/Byun, Matarić and others [34,55,58,65,81,105,112,115], represent robot environments by graphs. Nodes in such graphs correspond to distinct situations, places, or landmarks (such as doorways). They are connected by arcs if there exists a direct path between them.

Both approaches to robot mapping exhibit orthogonal strengths and weaknesses. Occupancy grids are easy to construct and to maintain in large-scale environments [9,102,103]. Occupancy grid approaches disambiguate different places based on the robot's geometric position within a global coordinate frame. The robot's position is estimated incrementally, based on odometric information and sensor readings taken by the robot. Thus, occupancy grid approaches usually use an unbounded number of sensor readings to determine a robot's location. To the extent that the position of a mobile robot can be tracked accurately [1], different positions for which sensors measurements look alike are naturally disambiguated. Nearby geometric places are recognized as such, even if the sensor measurements differ—which is often the case in dynamic environments where, e.g., humans can block a robot's sensors.

This is not the case for topological approaches. Topological approaches determine the position of the robot relative to the model primarily based on landmarks or distinct, momentary sensor features. For example, if the robot traverses two places that look alike, topological approaches often have difficulty determining if these places are the same or not (particularly if these places have been reached via different paths). Also, since sensory input usually depends strongly on the view-point of the robot, topological approaches may fail to recognize geometrically nearby places even in static environments, making it difficult to construct large-scale maps, particularly if sensor information is highly ambiguous.

On the other hand, grid-based approaches suffer from their enormous space and time complexity. This is because the resolution of a grid must be fine enough to capture every important detail of the world. The key advantage of topological representation is their compactness. The resolution of topological maps corresponds directly to the complexity of the environment. The compactness of topological representations gives them three key advantages over grid-based approaches: (a) they permit fast planning, (b) they facilitate interfacing to symbolic planners and problem-solvers, and (c) they provide more natural interfaces for human instructions (such as: *"go to room A"*). Since topological approaches usually do not require the exact determination of the geometric position of the robot, they often recover better from drift and slippage—phenomena that must constantly be monitored and compensated in grid-based approaches. To summarize, both paradigms have orthogonal strengths and weaknesses, which are summarized in Table 1.

This paper advocates the integration of both paradigms to gain the best of both worlds. The approach presented here combines grid-based (metric) and topological

[1] This paper presents a position tracking algorithm that works well in a restricted class of environments. This class includes the vast majority of indoor office environments.

Table 1: Advantages and disadvantages of grid-based and topological approaches to map building.

Grid-based (metric) approaches	Topological approaches
+ easy to build, represent, and maintain + recognition of places (based on geometry) is non-ambiguous and view point-independent + facilitates computation of shortest paths	+ permits efficient planning, low space complexity (resolution depends on the complexity of the environment) + does not require accurate determination of the robot's position + convenient representation for symbolic planner/problem solver, natural language
– planning inefficient, space-consuming (resolution does not depend on the complexity of the environment) – requires accurate determination of the robot's position – poor interface for most symbolic problem solvers	– difficult to construct and maintain in large-scale environments if sensor information is ambiguous – recognition of places often difficult, sensitive to the point of view – may yield suboptimal paths

representations. To construct a grid-based model of the environment, sensor values are interpreted by an artificial neural network and mapped into probabilities for occupancy. Multiple interpretations are integrated over time using Bayes's rule. On top of the grid representation, more compact topological maps are generated by splitting the metric map into coherent regions, separated through *critical lines*. Critical lines correspond to narrow passages such as doorways. By partitioning the metric map into a small number of regions, the number of topological entities is several orders of magnitude smaller than the number of cells in the grid representation. Therefore, the integration of both representations has unique advantages that cannot be found for either approach in isolation: the grid-based representation, which is easy to construct and maintain in environments of moderate size and complexity, models the world consistently and disambiguates different positions. The topological representation, which is grounded in the metric representation, facilitates fast planning and problem solving. The approach also inherits two disadvantages of grid-based approaches, namely the considerably larger memory requirements and the necessity for accurate localization.

The robots used in our research are shown in Figure 1. All robots are equipped with an array of sonar sensors, consisting of 24 or 16 sonars. Sonar sensors return

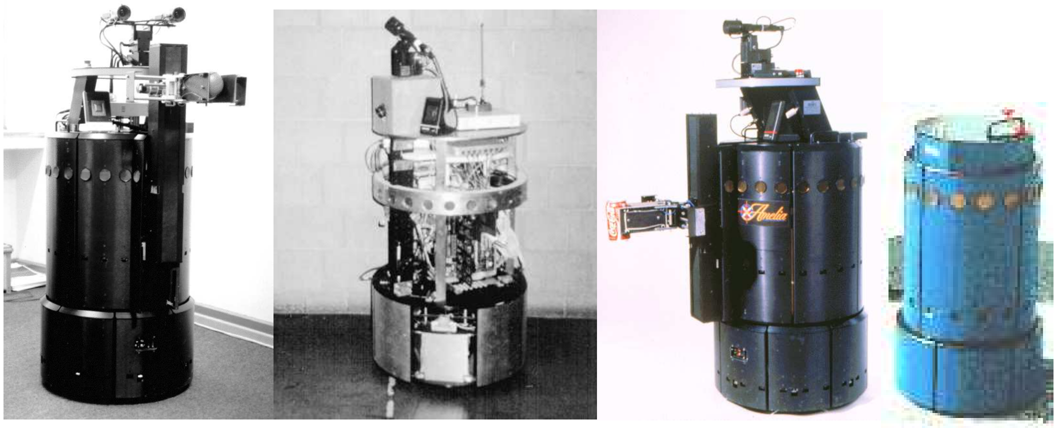


Fig. 1. The robots used in our research: RHINO (University of Bonn), XAVIER, and AMELIA (both CMU). The software has also been ported by RWI's B14 robots.

the proximity of surrounding obstacles, along with noise. One of these robots (AMELIA) is also equipped with a laser range finder, which measures proximity of nearby objects with higher spatial resolution. Throughout this paper, we will restrict ourselves to the interpretation of proximity sensors, although the methods described here have (in a prototype version) also been operated using cameras and infrared light sensors in addition to sonar sensors, using the image segmentation approach described in [9]. The integrated approach to map building has been tested extensively using sonar sensors in various indoor environments. It is now distributed commercially by a leading mobile robot manufacturer (Real world Interface, Inc.) as the sole navigation software along with their B14 and B21 robots.

The remainder of the paper is organized as follows: Section 2 describes our approach for building grid-based maps; followed by the description of our approach to building topological maps, described in Section 3; subsequently, Section 4 evaluates the utility of the integrated approach empirically; Section 5 reviews relevant literature; the paper is concluded by a discussion in Section 6.

2 Grid-Based Maps

The metric maps considered here are discrete, two-dimensional occupancy grids, as originally proposed in [31,70] and have since been implemented successfully in various systems. Each grid-cell $\langle x, y \rangle$ in a map has attached a value that measures the subjective belief that this cell is occupied. More specifically, it contains the belief as to whether or not the center of the robot can be moved to the center of that cell (it thus represents the *configuration space* of the robot projected into the x-y-plane, see e.g., [106,60]). Occupancy values are determined based on sensor readings.

This section describes the four major components of our approach to building grid-based maps [32,99]:

(i) **Interpretation.** Sensor readings are mapped to occupancy values.
(ii) **Integration.** Multiple sensor interpretations are integrated over time to yield a single, combined estimate of occupancy.
(iii) **Position estimation.** The position of the robot is continuously tracked and odometric errors are corrected.
(iv) **Exploration.** Shortest path through unoccupied regions are generated to move the robot greedily towards unexplored terrain.

Examples of metric maps are shown in various places in this paper.

2.1 Sensor Interpretation

To build metric maps, sensor reading must be "translated" into occupancy values $occ_{x,y}$ for each grid cell $\langle x, y\rangle$. The idea here is to train an artificial neural network [88] using Back-Propagation to map sonar measurements to occupancy values [99]. As shown in Figure 2, the input to the network consists of

– two values that encode $\langle x, y\rangle$ in polar coordinates relative to the robot (angle to the first of the four sensors, and distance), and
– the four sensor readings closest to $\langle x, y\rangle$.

The output target for the network is 1, if $\langle x, y\rangle$ is occupied, and 0 otherwise. Training examples can be obtained by operating a robot in a known environment, and recording its sensor readings. Notice that each sonar scan can be used to construct many training examples for different x-y coordinates. In our implementation, training examples are generated by a mobile robot simulator to facilitate the collection of the data. [2]

[2] Our simulator simulates sonar sensors in the following way: The main sonar cone—and only this cone is considered—is approximated by a set of five rays. For each ray, one of the following options is chosen at random: (1) a random short value is reported, (2) the

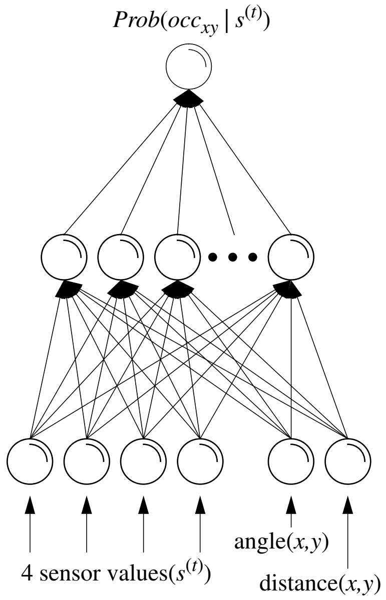


Fig. 2. An artificial neural network maps sensor measurements to probabilities of occupancy.

Once trained, the network generates values in $[0, 1]$ that can be interpreted as probability of occupancy. Formally, the interpretation of the network's output as conditional probability of a binary event is justified if the network is trained to minimize cross-entropy; we refer the interested reader to page 167 of Mitchell's textbook [67], which demonstrates that networks trained in this way approximate the maximum likelihood hypothesis [30,107].

Figure 3 shows three examples of sonar scans (top row, bird's eye view) along with their neural network interpretation (bottom row). The darker a value in the circular region around the robot, the larger the occupancy value computed by the network. Figures 3a and 3b show situations in a corridor. Here the network predicts the walls correctly. Notice the interpretation of the erroneous long reading in the left side of Figure 3a, and the erroneous short reading in 3b. For the area covered

correct distance is returned, and (3) a random large value is returned. The probability of these events depends on the angle between the ray and the surface normal of the obstacle. This model was adopted based on a series of empirical measurements [108]. The current simulator does not model (1) cones other than the main cone, (2) reflections involving more than one obstacle, (3) cross-sonar interference, and (4) temporal dependencies in sensor noise (cf. [56]). Nevertheless, the networks trained with our simulator generate good interpretations, as demonstrated throughout of this paper.

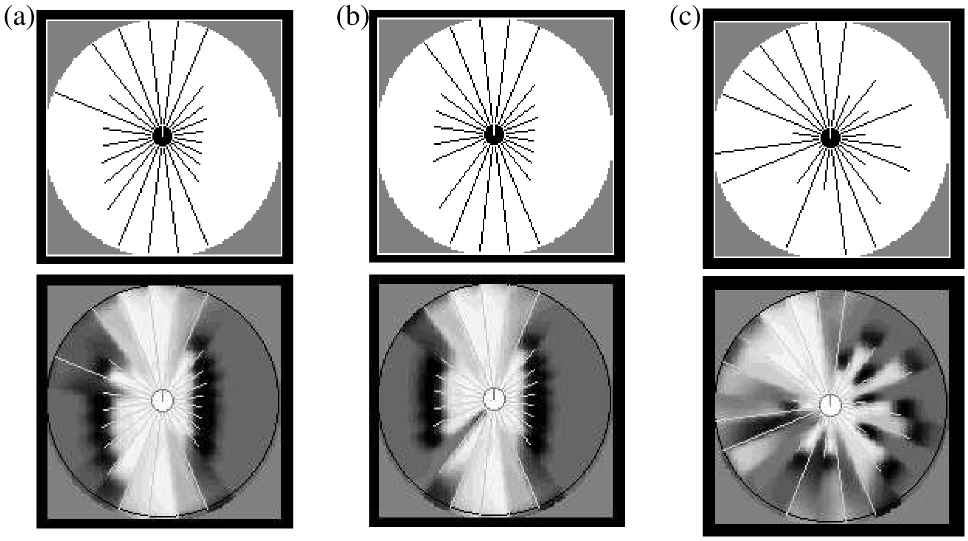


Fig. 3. Sensor interpretation: Three sample sonar scans (top row) and local occupancy maps (bottom row), as generated by the neural network. Bright regions indicate free-space, and dark regions indicate walls and obstacles (enlarged by a robot diameter).

by those readings, the network outputs roughly 0.5, which indicates maximum uncertainty. Figure 3c shows a different situation in which the interpretation of the sensor values is less straightforward. This example illustrates that the network interprets sensors in the context of neighboring sensors. Long readings are only interpreted as free-space if the neighboring sensors agree. Otherwise, the network returns values close to 0.5, which again indicates maximal uncertainty. Situations such as the one shown in Figure 3c—that defy simple interpretation—are typical for cluttered indoor environments.

Training a neural network to interpret sonar sensors has two key advantages over previous hand-crafted approaches to sensor interpretation:

(i) Since neural networks are trained based on examples, they can easily be adapted to new circumstances. For example, the walls in the competition ring of the 1994 AAAI robot competition [94] were much smoother than the walls in the building in which the software was originally developed. Even though time was short, the neural network could quickly be retrained to accommodate this new situation. Others, such as Pomerleau [82], also report a significant decrease in development time of integrated robotic systems through the use of machine learning algorithms.

(ii) Multiple sensor readings are interpreted simultaneously. To the best of our knowledge, all current approaches interpret each sensor reading individually, one at a time (see, e.g., [8,31,32,70]). Interpreting sensor readings in the context of their neighbors generally yields more accurate results. For example, the reflection properties of most surfaces depends strongly on the angle of the surface to the sonar beam, which can only be detected by interpreting multiple sonar sensors simultaneously. The neural networks take such phenomena into account for example, by ignoring long readings if their neighbors suggest that the angle between the sonar cone and the surface normal is large.

2.2 Integration Over Time

Sonar interpretations must be integrated over time to yield a single, consistent map. To do so, it is convenient to interpret the network's output for the t-th sensor reading (denoted by $s^{(t)}$) as the *probability* that a grid cell $\langle x, y\rangle$ is occupied conditioned on the sensor reading $s^{(t)}$:

$$Prob(occ_{x,y}|s^{(t)})$$

A map is obtained by integrating these probabilities for all available sensor readings, denoted by $s^{(1)}, s^{(2)}, \ldots, s^{(T)}$. In other words, the desired occupancy value for each grid cell $\langle x, y\rangle$ can be written as the probability

$$Prob(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}),$$

which is conditioned on *all* sensor readings. A straightforward approach to estimating this quantity is to apply Bayes's rule. To do so, one has to assume independence of the noise in different readings. More specifically, given the true occupancy of a grid cell $\langle x, y\rangle$, the conditional probability $Prob(s^{(t)}|occ_{x,y})$ must be assumed to be independent of $Prob(s^{(t')}|occ_{x,y})$ if $t \neq t'$. This assumption is not implausible—in fact, it is commonly made in approaches to building occupancy grids. It is important to note that the conditional independence assumption does not imply the independence of $Prob(s^{(t)})$ and $Prob(s^{(t')})$. The latter two random variables are usually dependent.

The desired probability can be computed in the following way:

$Prob(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}) =$

$$1 - \left(1 + \frac{Prob(occ_{x,y}|s^{(1)})}{1-Prob(occ_{x,y}|s^{(1)})} \prod_{\tau=2}^{T} \frac{Prob(occ_{x,y}|s^{(\tau)})}{1-Prob(occ_{x,y}|s^{(\tau)})} \frac{1-Prob(occ_{x,y})}{Prob(occ_{x,y})}\right)^{-1} \quad (1)$$

Here $Prob(occ_{x,y})$ denotes the prior probability for occupancy (which, if set to 0.5, can be omitted in this equation).

The update formula (1) follows directly from Bayes's rule and the conditional independence assumption. According to Bayes's rule,

$$\frac{Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T)})}{Prob(\neg occ_{x,y}|s^{(1)}, \ldots, s^{(T)})}$$
$$= \frac{Prob(s^{(T)}|occ_{x,y}, s^{(1)}, \ldots, s^{(T-1)})}{Prob(s^{(T)}|\neg occ_{x,y}, s^{(1)}, \ldots, s^{(T-1)})} \frac{Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}{Prob(\neg occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}$$

which can be simplified by virtue of the conditional independence assumption to

$$= \frac{Prob(s^{(T)}|occ_{x,y})}{Prob(s^{(T)}|\neg occ_{x,y})} \frac{Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}{Prob(\neg occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}.$$

Applying Bayes's rule to the first term leads to

$$= \frac{Prob(occ_{x,y}|s^{(T)})}{Prob(\neg occ_{x,y}|s^{(T)})} \frac{Prob(\neg occ_{x,y})}{Prob(occ_{x,y})} \frac{Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}{Prob(\neg occ_{x,y}|s^{(1)}, \ldots, s^{(T-1)})}$$

Induction over T yields:

$$= \frac{Prob(occ_{x,y})}{1-Prob(occ_{x,y})} \prod_{\tau=1}^{T} \frac{Prob(occ_{x,y}|s^{(\tau)})}{1-Prob(occ_{x,y}|s^{(\tau)})} \frac{1-Prob(occ_{x,y})}{Prob(occ_{x,y})} \quad (2)$$

The update equation (1) is now obtained by solving (2) for $Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T)})$, using the fact that $Prob(\neg occ_{x,y}|s^{(1)}, \ldots, s^{(T)}) = 1 - Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T)})$. This probabilistic update rule, which is sound given our conditional independence assumption, is frequently used for the accumulation of sensor evidence [70,79]. It differs from Bayes networks [79] in that albeit the fact that occupancy causally determines sensor readings $\{s^{(\tau)}\}_{\tau=1,\ldots,T}$ and not the other way round, the networks represent the inverse conditional probability: $Prob(occ_{x,y}|s^{(t)})$. Notice that Equation (1) can be used to update occupancy values *incrementally*, i.e., at any point in time τ it suffices to memorize a single value per grid cell: $Prob(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(\tau)})$. Technically speaking, this single value is a *sufficient statistic* for $s^{(1)}, s^{(2)}, \ldots, s^{(\tau)}$ [107].

Figure 4 shows an example map. Although this figure appears to be three-dimensional, the actual map is two-dimensional: The higher a point, the more likely it is to be occupied. This map was using a simulator in order to investigate map building in the absence of odometric errors—all other maps shown in this paper were constructed using a real robot. In this particular run, the simulator did not introduce odometric errors; it did, however, model noise in perception. The map shown in Figure 4 is approximately 105 by 63 meters in size and was acquired in 45 minutes of autonomous robot exploration. The algorithm used for exploration is described below. As can be seen in Figure 4, the neural network approach to interpreting sensor data combined with the probabilistic method for integrating them yield considerably more accurate maps. In the presence of odometric errors maps are usually less accurate.

2.3 Localization

The accuracy of the metric map depends crucially on the alignment of the robot with its map. Unfortunately, slippage and drift can have devastating effects on the estimation of the robot position. Identifying and correcting for slippage and drift (odometric error) is therefore an important issue in map building [7,22,85].

Figures 5 and 10 give examples that illustrate the importance of position estimation in grid-based robot mapping. For example, in Figure 5a the position is determined solely based on dead-reckoning. After approximately 15 minutes of robot operation, the position error is approximately 11.5 meters. Obviously, the resulting map is too erroneous to be of practical use. The map shown in Figure 5b, constructed from the identical data, is the result of exploiting and integrating three sources of information:

(i) **Wheel encoders.** Wheel encoders measure the revolution of the robot's wheels. Based on their measurements, odometry yields an estimate of the robot's position at any point in time. We will denote this estimate by $\langle x^*_{\text{robot}}, y^*_{\text{robot}}, \theta^*_{\text{robot}}\rangle$. As can be seen from Figure 5a and 10b, odometry is very accurate over short time intervals, but inaccurate in the long run.
(ii) **Map matching.** Whenever the robot interprets an actual sensor reading, it constructs a "local" map (such as the ones shown in Figure 3). The *correlation*

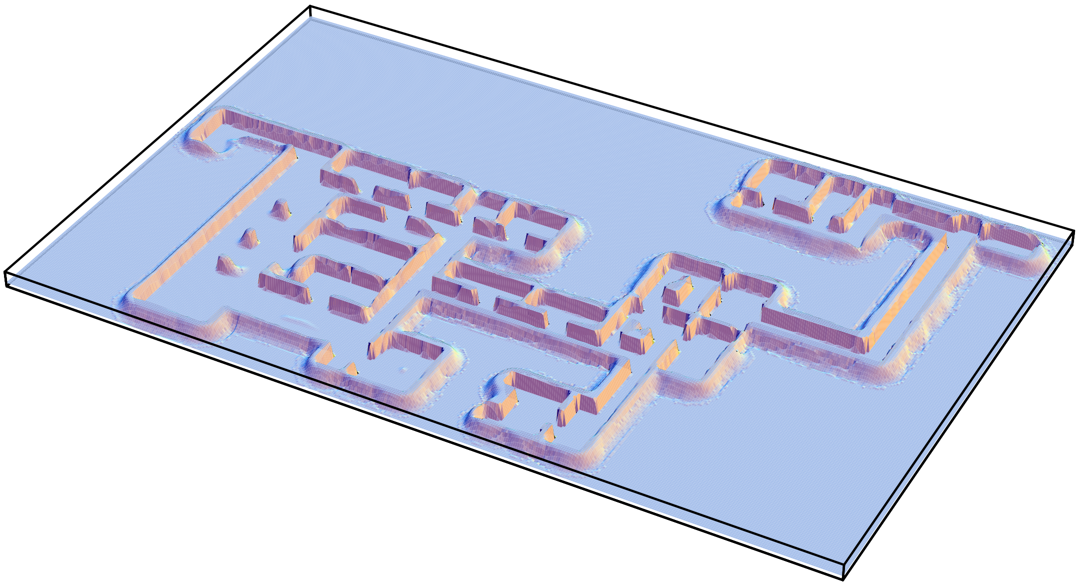
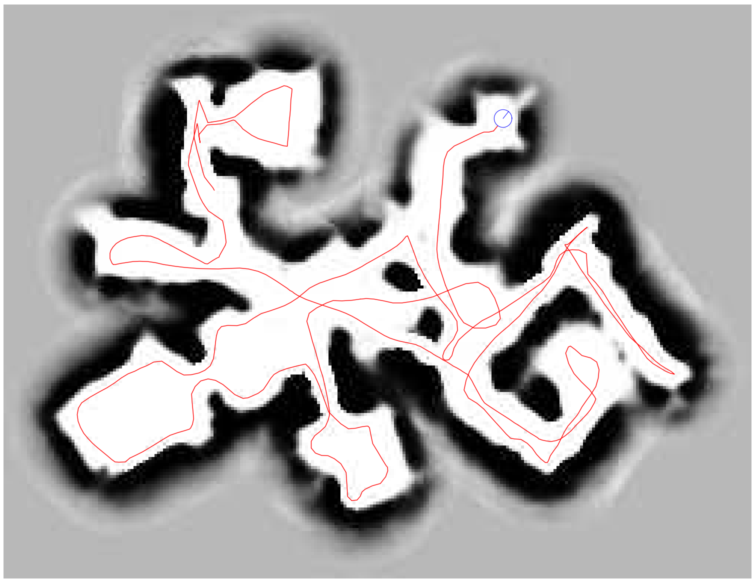


Fig. 4. If there were no odometric errors, large maps could be built with ease. Shown here is a map of the size 105 by 63 meters, which the robot learned fully autonomously in approximately 45 minutes, using the map building and exploration algorithms described in this paper. The higher a point in the map, the more likely it is to be occupied. This map has been built using a simulator. The simulator uses a fairly realistic sonar noise model, but here it does not model odometric error.

of the local and the corresponding section of the global map is a measure of their correspondence. Obviously, the more correlated both maps are, the more alike the local map looks to the global one, hence the more plausible it is. The correlation is a function of the robot's position $\langle x^*_{\text{robot}}, y^*_{\text{robot}}, \theta^*_{\text{robot}}\rangle$. Thus, the correlation of the local with the global map gives a second source of information for aligning the robot's position.

Technically, local maps are computed in local, robot-centered coordinates, whereas global maps are computed in a global coordinate system. As a result, each grid cell of the global map that overlaps with the local map, overlaps almost always with exactly four grid cells of the local map, as shown in Figure 7. Let $\langle x, y\rangle$ be the coordinates of a cell in the global map which overlaps with the local map, and let $\langle x', y'\rangle$ denote the corresponding coordinates in the local map. Let $\langle x_i, y_i\rangle$ with $i = 1, \ldots, 4$ denote the coordinates of the four grid points in the local map that are nearest to $\langle x', y'\rangle$ (which are unique with probability 1). The global occupancy occ_{xy} is then matched with the local occupancy value obtained using the following interpolation:

(a)



(b)

Fig. 5. Map constructed (a) without and (b) with the position estimation mechanism described in this paper. In (a), only the wheel encoders are used to determine the robot's position. The positional error accumulates to more than 11 meters, and the resulting map is clearly unusable. This illustrates the importance of sensor-based position estimation for map building.

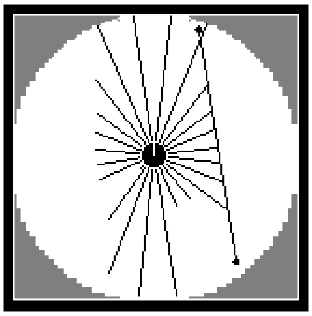


Fig. 6. Wall, detected by considering five adjacent sonar measurements. Wall orientations are used to correct for dead-reckoning errors in the robot orientation θ_{robot}.

$$\frac{\sum_{i=1}^{4} |x - x_i| \; |y - y_i| \; lococc_{x_i y_i}}{\sum_{i=1}^{4} |x - x_i| \; |y - y_i|} \,. \tag{3}$$

where $lococc$ denotes the local occupancy grid. In other words, the coordinate of a global grid cell is projected into the local robot's coordinates, and the local occupancy value is obtained by interpolation. The interpolating function is similar in spirit to Shepard's interpolation [93]. It has several interesting properties, most notably it is smooth (continuous) and almost everywhere differentiable in $\langle x^*_{\text{robot}}, y^*_{\text{robot}}, \theta^*_{\text{robot}} \rangle$.

The key advantage of interpolating between occupancy values, instead of simply picking the nearest one, lies in the fact that gradient ascent can be employed to maximize the correlation between the global and the local map. The correlation of the two maps is a differentiable function of the interpolated local map values, which themselves are differentiable functions of the robot's coordinates. Thus, given the interpolation described here the correlation function is differentiable in the robot's position. Gradient ascent is an efficient search scheme for searching large spaces, which usually suffers from the danger of local minima. If the robot's error is small, which is typically the case when tracking a robot's position(since odometric errors are small), gradient ascent search usually leads to the correct solution.

(iii) **Wall orientation.** A third component memorizes the *global wall orientation* [25,42]. This approach rests on the restrictive assumption that walls are either parallel or orthogonal to each other or differ by more than 15 degrees from these canonical wall directions. In the beginning of map building, the global orientation of walls is estimated by analyzing consecutive sonar scans (cf.

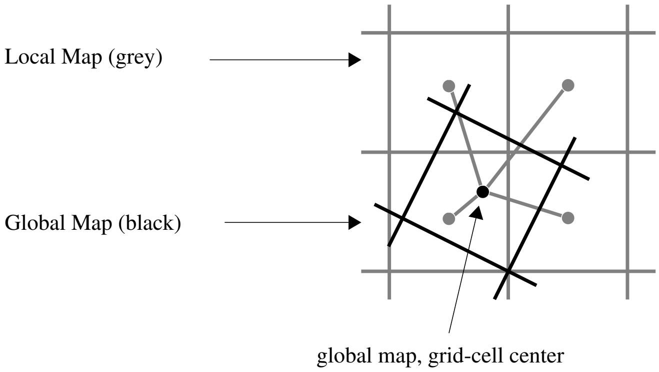


Fig. 7. Map matching. Since local maps are recorded in local robot coordinates and global maps are recorded in global coordinates, each cell in the global occupancy grid usually overlaps with four local grid cell. The values of these four cells are interpolated to yield a single occupancy value.

Figure 6). This is done by searching straight lines that connect the endpoints of five or more adjacent sonar measurements. Once the global wall orientation (denoted by θ_{wall}) has been determined, subsequent sonar scans are used to realign the robot's orientation. More specifically, suppose the robot detects a line in a sonar scan. Let $\hat{\theta}$ be the angle of this line relative to the robot. Then—in the ideal case—

$$\alpha(\theta_{\text{robot}}, \hat{\theta}, \theta_{\text{wall}}) := (\theta_{\text{robot}} + \hat{\theta} - \theta_{\text{wall}}) \;\; \textit{modulo} \;\; 90^\circ$$

should be zero, i.e., the detected wall should be orthogonal or parallel to θ_{wall}. If this is not the case, the robot corrects its orientation accordingly, by maximizing

$$\sigma(\alpha) := \begin{cases} (|\alpha| - 15^\circ)^2 & \text{if } |\alpha| \leq 15^\circ \\ 0 & \text{if } |\alpha| > 15^\circ \end{cases}$$

using gradient ascent. The function σ and its first derivative is shown in Figure 8. Walls whose orientation differ from the global wall orientation by 15° or more have no effect. This is because the derivative of σ with respect to θ_{robot} is zero if $|\alpha| \geq 15^\circ$. If $|\alpha|$ is smaller than 15°, the gradient of σ with respect to θ_{robot} is non-zero and increases linearly as α approaches 0°. Thus, the more similar the global and the observed wall orientation, the larger the gradient, and the stronger the effect on the estimated robot orientation θ_{robot}. This graded scheme was empirically found to yield the best results in various populated

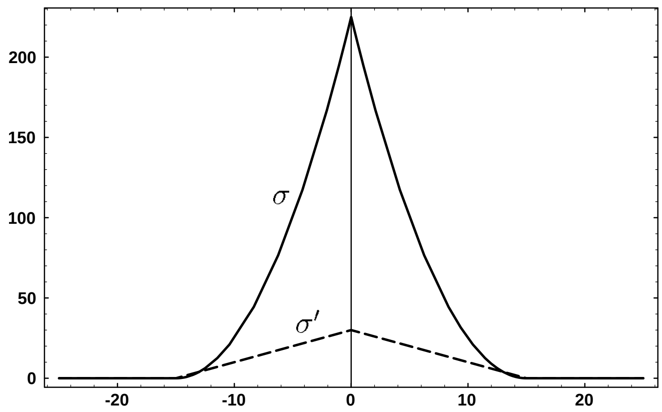


Fig. 8. The function σ and its derivative. σ is most sensitive to values close to zero. Thus, small deviations of the expected and observed wall orientation have the strongest effect. If this deviation is larger than 15°, it is completely ignored. Consequently, walls that deviate from the expected wall orientation by more than 15° have no effect.

---

indoor environments. In particular, it was found to be robust to noise, errors in wall detection, obstacles such as desks and chairs, and people walking by.

The exact function that is being minimized when calculating the robot's position is:

$$
\begin{aligned}
J := \beta_1 & \left[(x^*_{\text{robot}} - x_{\text{robot}})^2 + (y^*_{\text{robot}} - y_{\text{robot}})^2 \right] \\
& + \beta_2 \, (\theta^*_{\text{robot}} - \theta_{\text{robot}})^2 \\
& - \beta_3 \, corr(x_{\text{robot}}, y_{\text{robot}}, \theta_{\text{robot}}) \\
& - \beta_4 \, \sigma(\alpha(\theta_{\text{robot}}, \hat{\theta}, \theta_{\text{wall}}))
\end{aligned}
\tag{4}
$$

Here β_1, β_2, β_3, and β_4 are gain parameters that trade off the different sources of information. The first two terms in (4) correspond to the odometry of the robot. The third term measures the correlation between the global and the local map, and the fourth term relates the global wall orientation to the observed wall orientation. Equation (4) is differentiable, and gradient descent is employed to minimize J. Gradient descent is an iterative search scheme, whose accuracy usually increases with the number of iterations. When a new sonar reading arrives, the previous gradient search is terminated and its result is incorporated into the current position estimation. Consequently, the position tracking algorithm is an anytime algorithm [28] whose accuracy depends on the available computation time.

An example map of a competition ring constructed at the 1994 AAAI autonomous

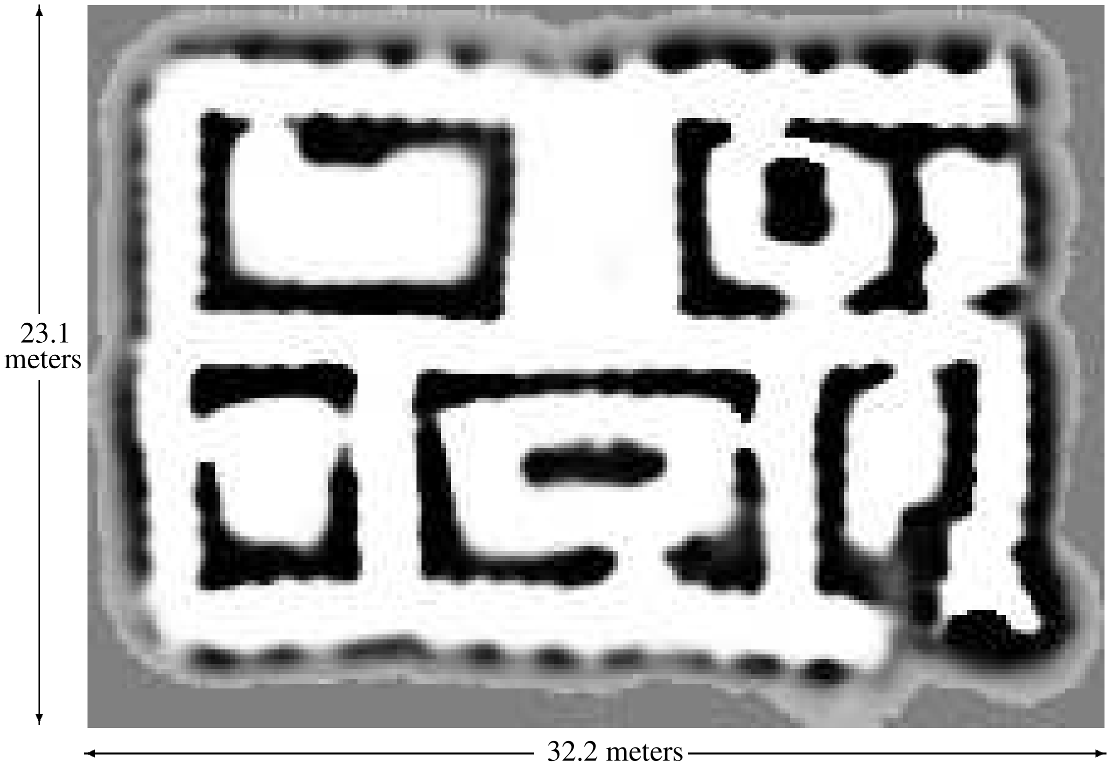


Fig. 9. Grid-based map, constructed at the 1994 AAAI autonomous mobile robot competition with the techniques described here.

robot competition is shown in Figure 9. This map contains an open area, which was intentionally created by the competition organizers to test the robot's ability to navigate in large open spaces [94]. In [103], occupancy maps are constructed using stereo vision for depth estimation [37,38]. As shown there, sonar and stereo information have somewhat orthogonal sensor characteristics <and thus can complement each other.

Position control based on odometry and map correlation alone (items 1 and 2 above) works well if the robot travels through mapped terrain [99], but fails to localize the robot if it explores and maps unknown terrain. The third mechanism, which arguably relies on a restrictive assumption concerning the nature of indoor environments, has proven extremely valuable when autonomously exploring and mapping large-scale indoor environments. Notice that all maps shown in this paper (with the exception of the maps shown in Figures 5a and 10b) have been generated using this position estimation mechanisms.

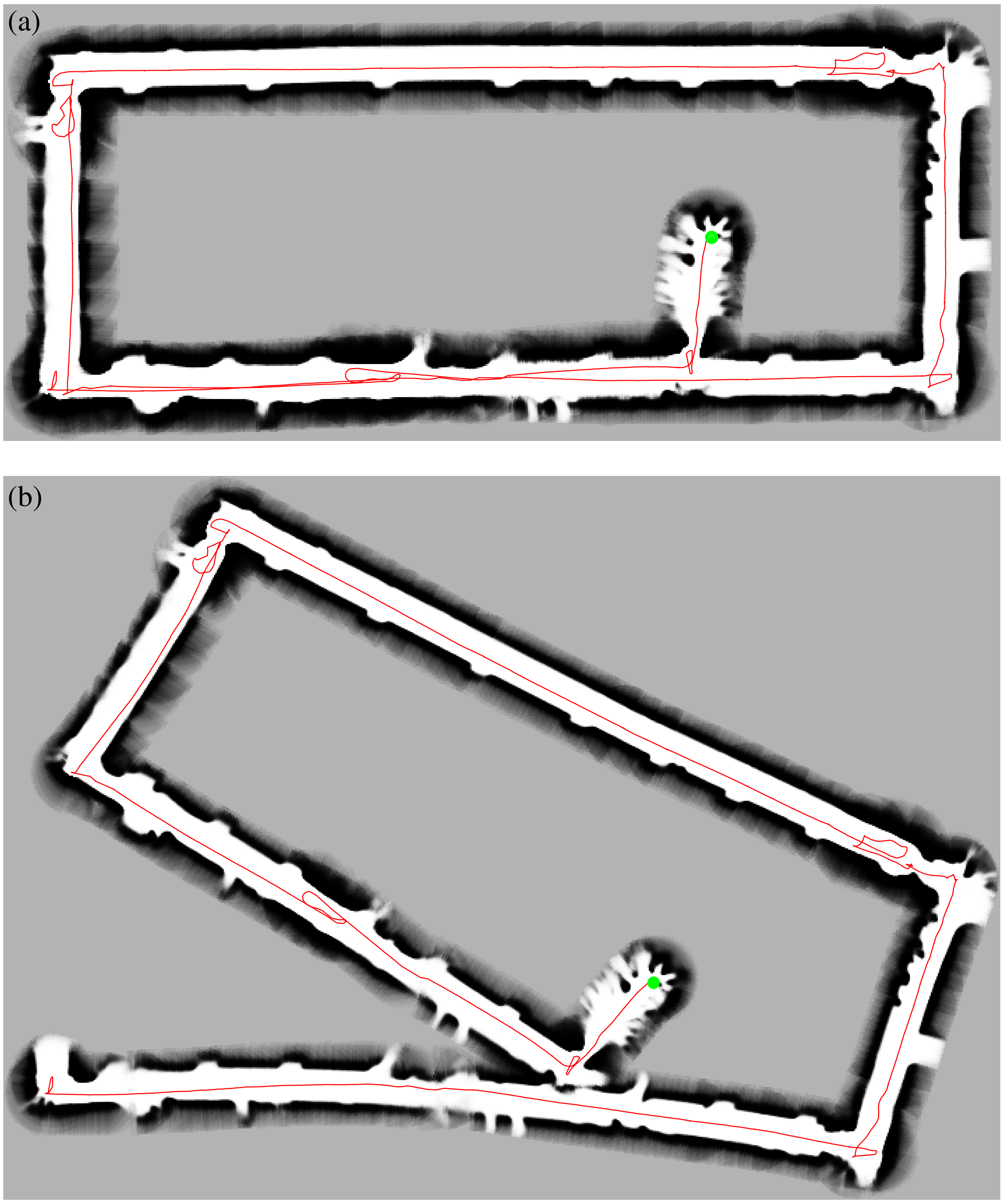


Fig. 10. (a) The challenge: A cyclic map of the size 60 by 20 meters, built using a laser range finder (instead of sonar sensors). (b) A map obtained from the same data without position control. Some of the floor in the testing environment is made of tiles, which introduces significant error in the robot's odometry. Most existing topological map building methods should have great difficulty building such maps, since most sensor measurements look alike and are highly ambiguous.

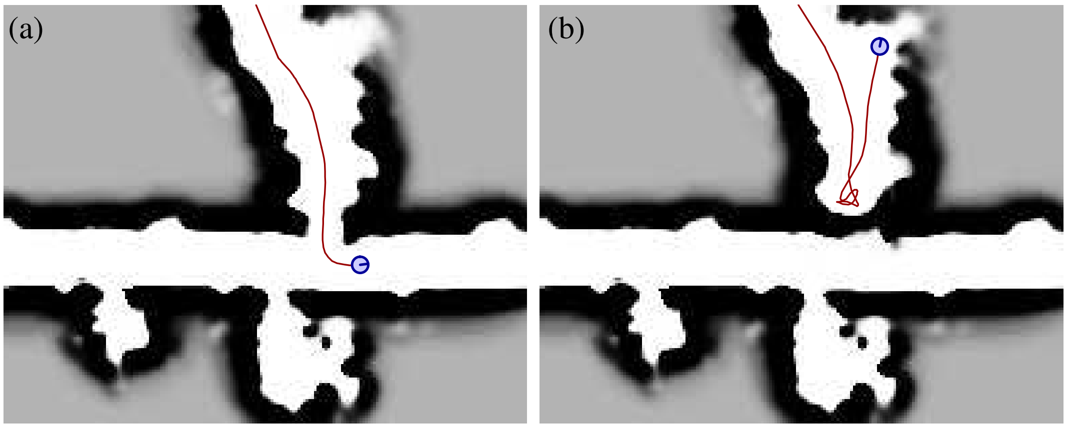


Fig. 11. A dynamic environment. (a) The robot moves through an open door. (b) The door is now close, in response to which the robot changes its map and takes a different route.

A difficult challenge for any localization method—and in fact any approach for learning maps—is a large circular environment, where local sensor readings or short histories thereof are insufficient to disambiguate the robot's location. Figure 10 depicts such an environment. Here the robot starts in a corner of a circular corridor, and after traversing more than 160m of unmapped corridor it reaches the same location. The map shown in Figure 10 has been obtained using a laser range finder to measure proximity. The laser range finder is more accurate and has an increased angular resolution (0.5° instead of 15°), which leads to more accurate localization results. In nine experiments using sonar and laser sensors, laser was found to generate an accurate map in five out of five cases, whereas sonar failed in two out of four cases, rendering maps that were slightly too erroneous to be of practical use (1m translational error). Figure 10b depicts a map constructed without the position correction mechanisms described here.

2.4 Dynamic Environments

Our basic approach to sensor integration assumes that the world is static. In particular, sensor readings at any two different points in time τ and τ' have the same weight in determining $Prob(occ_{x,y}|s^{(1)},\ldots,s^{(T)})$, even if τ precedes τ' by a large temporal margin. Intuitively, in dynamic environments more recent readings carry more information than more distant ones. This intuition can be incorporated by decaying the influence of individual sensor readings exponentially over time. Let γ

with $0 \leq \gamma \leq 1$ be the *decay factor*.

$$Prob(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}) =$$

$$1 - \left(1 + \gamma^{T-1} \frac{Prob(occ_{x,y}|s^{(1)})}{1-Prob(occ_{x,y}|s^{(1)})} \prod_{\tau=2}^{T} \gamma^{T-\tau} \frac{Prob(occ_{x,y}|s^{(\tau)})}{1-Prob(occ_{x,y}|s^{(\tau)})} \frac{1-Prob(occ_{x,y})}{Prob(occ_{x,y})}\right)^{-1}$$

This modified update rule (cf. (1)) weighs more recent sensor readings exponentially stronger than more distant ones. If γ is chosen appropriately, this approach prevents $Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T)})$ from taking its extreme values. While in theory, the occupancy probability $Prob(occ_{x,y}|s^{(1)}, \ldots, s^{(T)})$ may never attain its extreme values zero or one, in practice the occupancy values may do so due to the limited numerical resolution of digital computers. It is generally desirable to avoid these extreme values, since they suggest that a robot is absolutely certain about the occupancy of a grid cell—which a robot never can be due to the non-deterministic nature of its sensors.

Figure 11 depicts results obtained in a changing environment. Here the robot re-uses a previously built map for navigating from our lab into the corridor. In Figure 11b, the door is closed. After acquiring some evidence that the door is closed, indicating by the circular motion in front of the door shown in Figure 11b, the model is revised and the planning routines (described below) change the motion direction accordingly. Here γ is set to 0.9999. This decay factor is used in all our experiments, even if the environment is known to be static.

The reader may notice that our approach does not fully model dynamic environments; instead, it merely adapts to changes. For example, our approach is incapable of detecting dynamic regularities in the environment, such as doors, which are either open or closes and which change their status perpetually. As a consequence, once a door has been recognized as being closed it will be assumed to be closed until the robot receives evidence to the contrary. Modeling dynamic environments using metric representations is largely an open research area. Schneider, in his M.Sc. thesis [91], has extended our approach to detect certain types of regularities. His approach analyzed the *variance* and the *auto-correlation* of interpretations over time, enabling it to reliably identify and label regions in the map whose occupancy changed regularly over time. His approach was successfully applied to detecting dynamic objects at static locations (such as doors); however, it is not able to model moving objects such as humans.

To autonomously acquire maps, the robot has to explore. The idea for (greedy) exploration is to let the robot always move on a minimum-cost path to the nearest unexplored grid cell. The cost for traversing a grid cell is determined by its occupancy value. The minimum-cost path is computed using a modified version of *value iteration*, a popular dynamic programming algorithm [4,44,83]:

(i) **Initialization.** Unexplored grid cells are initialized with 0, explored ones with ∞:

$$V_{x,y} \longleftarrow \begin{cases} 0, & \text{if } \langle x, y \rangle \text{ unexplored} \\ \infty, & \text{if } \langle x, y \rangle \text{ explored} \end{cases}$$

Grid cells are considered *explored* if their occupancy value $Prob(occ_{x,y})$ has been updated at least once. Otherwise, they are *unexplored*.

(ii) **Update loop.** For all explored grid cells $\langle x, y \rangle$ do:

$$V_{x,y} \longleftarrow \min_{\substack{\xi=-1,0,1 \\ \zeta=-1,0,1}} \left\{ V_{x+\xi,y+\zeta} \quad + \quad Prob(occ_{x+\xi,y+\zeta}) \right\}$$

Value iteration updates the value of all explored grid cells by the value of their best neighbors, plus the costs of moving to this neighbor (just like A* [75] or Dijkstra's famous shortest path algorithm [6]). Cost is here equivalent to the probability $Prob(occ_{x,y})$ that a grid cell $\langle x, y \rangle$ is occupied. The update rule is iterated. When the update converges, each value $V_{x,y}$ measures the *cumulative cost* for moving to the nearest unexplored cell. However, control can be generated at any time [28], long before value iteration converges. in the worst case, the computation of V requires $O(n^2)$ steps with n being the number of grid cells.

(iii) **Determine motion direction.** To determine where to explore next, the robot generates a minimum-cost path to the unexplored. This is done by steepest descent in V, starting at the actual robot position. Determining the motion direction is computationally cheap; it is the computation of V which requires significant computation. Determining the motion direction is done in regular time intervals and is fully interleaved with updating V.

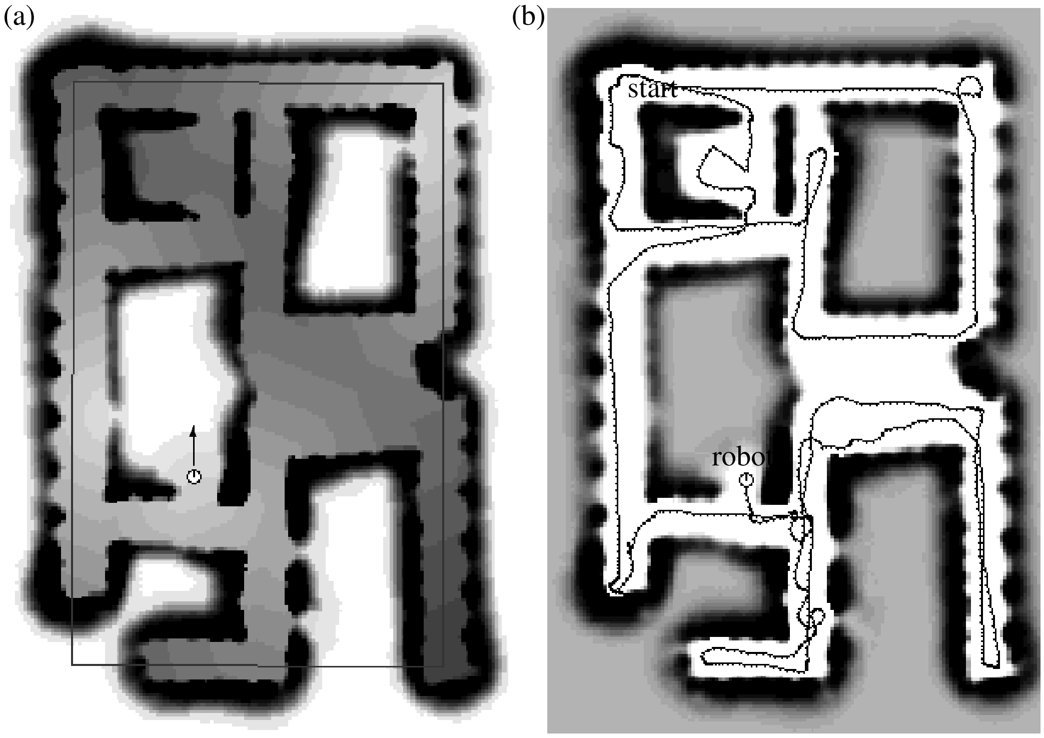


Fig. 12. **Autonomous exploration.** (a) Exploration values V, computed by value iteration. White regions are completely unexplored. By following the grey-scale gradient, the robot moves to the next unexplored area on a minimum-cost path. The large black rectangle indicates the global wall orientation θ_{wall}. (b) Actual path traveled during autonomous exploration, along with the resulting metric map.

Figure 12a shows V after convergence using the map shown in Figure 12b. All white regions are unexplored, and the grey-level indicates the cumulative costs V for moving towards the nearest unexplored point. Notice that all minima of the value function correspond to unexplored regions—there are no local minima. For every point $\langle x, y \rangle$, steepest descent in V leads to the nearest unexplored area.

Unfortunately, plain value iteration is too inefficient to allow the robot to explore in real-time. Strictly speaking, the basic value iteration algorithm can only be applied if the cost function does not increase (which frequently happens when the map is updated). This is because if the cost function increases, previously adjusted values V might become too small. While value iteration quickly decreases values that are too large, *increasing* values can be arbitrarily slow [99]. Consequently, the basic value iteration algorithm requires that the value function be initialized completely (Step i) whenever the map—and thus the cost function—is updated. This is very inefficient, since the map is updated almost constantly. To avoid complete re-initializations, and to further increase the efficiency of the approach, the basic algorithm was extended in the following way:

(iv) **Selective reset phase.** Every time the map is updated, only values $V_{x,y}$ that are too small are identified and reset. This is achieved by the following loop, which is iterated:

For all explored $\langle x, y \rangle$ do:

$$V_{x,y} \longleftarrow \infty \quad \text{if} \quad V_{x,y} < \min_{\substack{\xi=-1,0,1 \\ \zeta=-1,0,1}} \left\{ V_{x+\xi,y+\zeta} \;+\; Prob(occ_{x+\xi,y+\zeta}) \right\}$$

Notice that the remaining $V_{x,y}$-values are not affected. Resetting the value table in this way bears close resemblance to the value iteration algorithm described above.

(v) **Bounding box.** To focus value iteration, a rectangular bounding box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is maintained that contains all grid cells in which $V_{x,y}$ may change. This box is easily maintained in the value iteration update. As a result, value iteration focuses only on a small fraction of the grid, hence it converges much faster.

Notice that the bounding box bears some similarity to *prioritized sweeping* [69]. Prioritized sweeping determines the order of updates according the expected benefit of each particular update. Bounding boxes are a cheap implementation of the same idea. Their advantage lies in the fact that rectangular arrays can be processed very efficiently; however, they are less selective, which typically increases the total number of updates when compared to prioritized sweeping.

Figure 12 shows a snapshot of autonomous exploration in the environment depicted in Figure 9. Figure 12a shows the value function after convergence. All white regions are unexplored, and the grey-level indicates the cumulative costs for moving towards the nearest unexplored point. Notice that all minima of the value function correspond to unexplored regions—there are no local minima. Once value iteration converges, greedy exploration simply amounts to steepest descent in the value function, which can be done very efficiently. The right plot, Figure 12b, sketches the path taken during autonomous exploration. At the current point, the robot has already explored the major hallways, and is about to continue exploration of a room. Circular motion, such as found in the bottom of this plot, occurs when two unexplored regions are about equally far away (=same costs) or when the planner has not yet converged. Notice that the complete exploration run shown here took less than 15 minutes. The robot moved constantly, and frequently reached a velocity of 80 to 90 cm/sec (see also [9,36,103]). The exploration of the map shown in Figure 4 required approximately 45 minutes.

Value iteration is a very general procedure which has several properties that make it attractive for real-time mobile robot navigation:

- **Any-time algorithm.** As mentioned above, value iteration can be used as an any-time planner [28]. Any-time algorithms are able to make decisions regardless of the time spent for computation. The more time that is available, however, the better the results. Value iteration allows the robot to explore in real-time.
- **Full exception handling.** Value iteration pre-plans for arbitrary robot locations. This is because V is computed for every location in the map, not just the current location of the robot. Consequently, the robot can quickly react if it finds itself to be in an unexpected location and generate appropriate motion directions without any additional computational effort. This is particularly important in our approach, since the robot uses a fast routine for avoiding collisions with obstacles, which may modify the motion direction commanded by the planner at its own whim [36].
- **Multi-agent exploration.** Since value iteration generates values for all grid-cells, it can easily be used for collaborative multi-agent exploration.
- **Point-to-point navigation.** By changing the initialization of V (Step i), the same approach is used for point-to-point navigation [99].

In grid maps of the size 30 by 30 meters, optimized value iteration, done from scratch, requires approximately 2 to 10 seconds on a SUN Sparc station. Planning point-to-point navigation from scratch using the map shown in Figure 4, which due to its many small corridors poses a difficult real-time planning problem, requires up to 15 seconds depending on the location of the target point(s). In cases where the selective reset step does not reset large fractions of the map (which is the common situation), value iteration converges in less than a second for the size maps shown here. For example, the planning time in the map shown in Fig. 9 lies typically under a tenth of a second. In the light of these results, one might be inclined to think that grid-based maps are sufficient for autonomous robot navigation. However, value iteration (and similar planning approaches) requires time quadratic in the number of grid cells, imposing intrinsic scaling limitations that prohibit efficient planning in large-scale domains. Due to their compactness, topological maps scale much better to large environments. In what follows we will describe our approach for deriving topological graphs from grid maps.

3 Topological Maps

3.1 Constructing Topological Maps

Topological maps are built on top of the grid-based maps. The key idea, which is visualized in Figure 13, is simple but effective: Grid-based maps are decomposed into a small set of regions separated by narrow passages such as doorways. These narrow passages, which are called *critical lines*, are found by analyzing a skeleton of the environment. The partitioned map is mapped into an isomorphic graph, where nodes correspond to regions and arcs connect neighboring regions. This graph is the topological map.

The precise algorithm works as follows:

(i) **Thresholding.** Initially, each occupancy value in the occupancy grid is thresholded. Cells whose occupancy value is below the threshold are considered free-space (denoted by C). All other points are considered occupied (denoted by $\bar{C}$).

(ii) **Voronoi diagram.** For each point in free-space $\langle x, y\rangle \in C$, there is one or more *nearest point(s)* in the occupied space $\bar{C}$. We will call these points the *basis points of* $\langle x, y\rangle$, and the distance between $\langle x, y\rangle$ and its basis points the *clearance of* $\langle x, y\rangle$. The Voronoi diagram [16,58,60,77] is the set of points in free-space that have at least two different (equidistant) basis-points. Figure 13b depicts a Voronoi diagram.

(iii) **Critical points.** The key idea for partitioning the free-space is to find *"critical points."* Critical points $\langle x, y\rangle$ are points on the Voronoi diagram that minimize clearance locally. In other words, each critical point $\langle x, y\rangle$ has the following two properties: (a) it is part of the Voronoi diagram, and (b) the clearance of all points in an ε-neighborhood of $\langle x, y\rangle$ is *not* smaller. Figure 13c illustrates critical points.

(iv) **Critical lines.** Critical lines are obtained by connecting each critical point with its basis points (cf. Figure 13d). Critical points have exactly two basis points (otherwise they would not be local minima of the clearance function). Critical lines partition the free-space into disjoint regions (see also Figure 13e).

(v) **Topological graph**. The partitioning is mapped into an isomorphic graph.

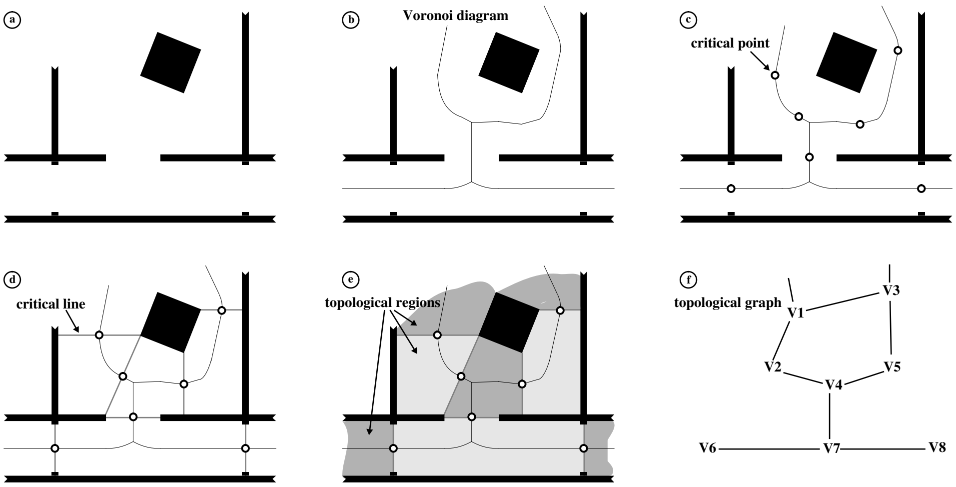


Fig. 13. **Extracting topological maps.** (a) Metric map, (b) Voronoi diagram, (c) critical points, (d) critical lines, (e) topological regions, and (f) the topological graph.

Each region corresponds to a node in the topological graph, and each critical line to an arc. Figure 13f shows an example of a topological graph.

The particular definition of critical lines for decomposing the metric map is motivated by two observations. First, when passing through a critical line, the robot is forced to move through a region that is considerably narrow, when compared to the neighboring regions. Hence, the loss in performance inferred by planning using the topological map (as opposed to the grid-based map) is considerably smaller. Secondly, narrow regions are more likely blocked by obstacles (such as doors, which can be open or closed). The reader should note that these arguments are somewhat intuitive, as they are not backed up with mathematical proofs.

Figure 14 illustrates the topological map extracted from the grid-based map depicted in Figure 9. Figure 14a shows the Voronoi diagram of the thresholded map, and Figure 14b depicts the critical lines (the critical points are on the intersections of critical lines and the Voronoi diagram). The resulting partitioning and the topological graph are shown in Figures 14c and 14d. As can be seen, the free-space has been partitioned into 67 regions. Additional examples of metric and topological maps are shown in Figures 15 and 16. These maps are partitioned into 22 (Figures 15c and 15d) and 39 regions (Figures 16c and 16d).

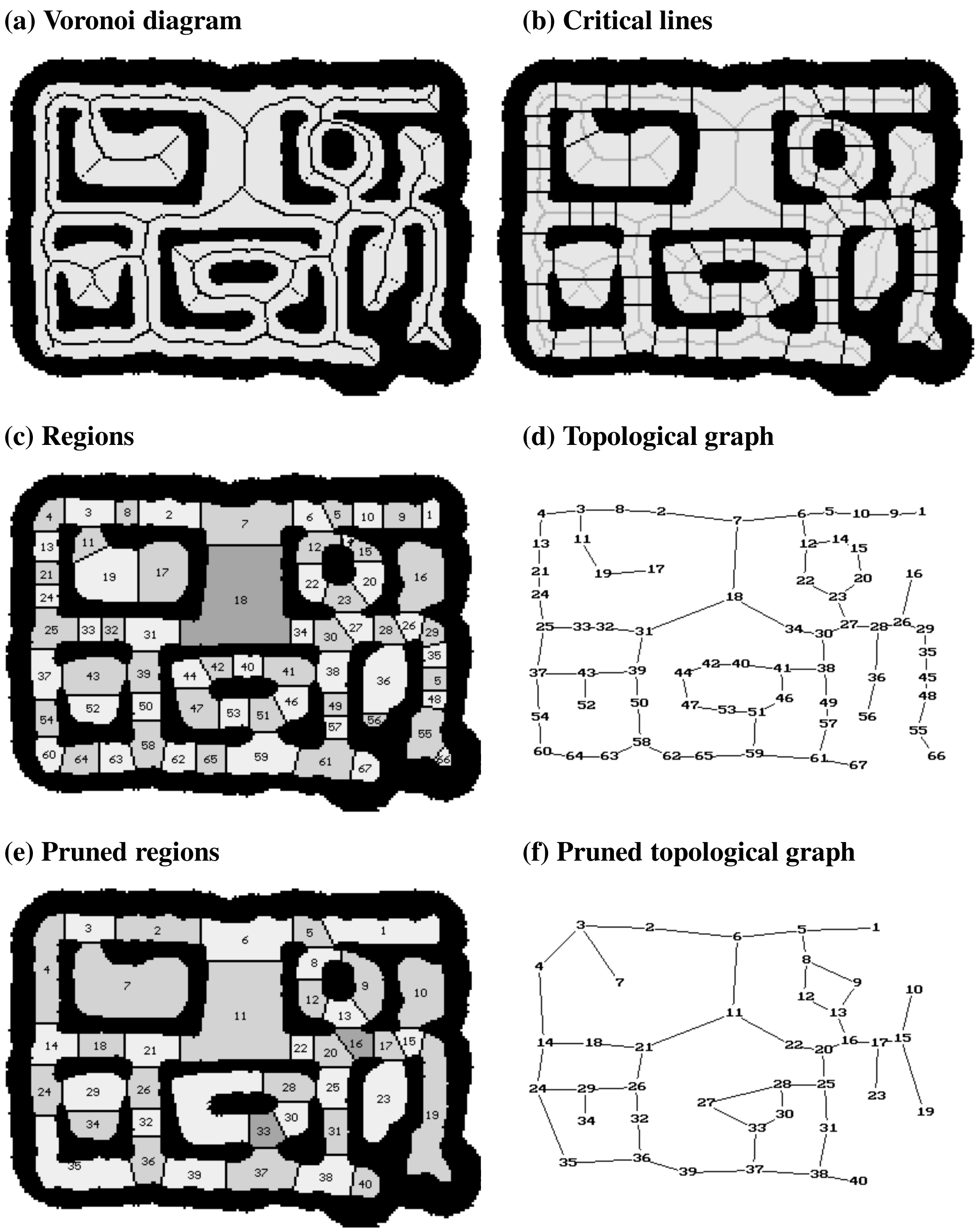


Fig. 14. Extracting the topological graph from the map depicted in Figure 9: (a) Voronoi diagram, (b) Critical points and lines, (c) regions, and (d) the final graph. (e) and (f) show a pruned version (see text).

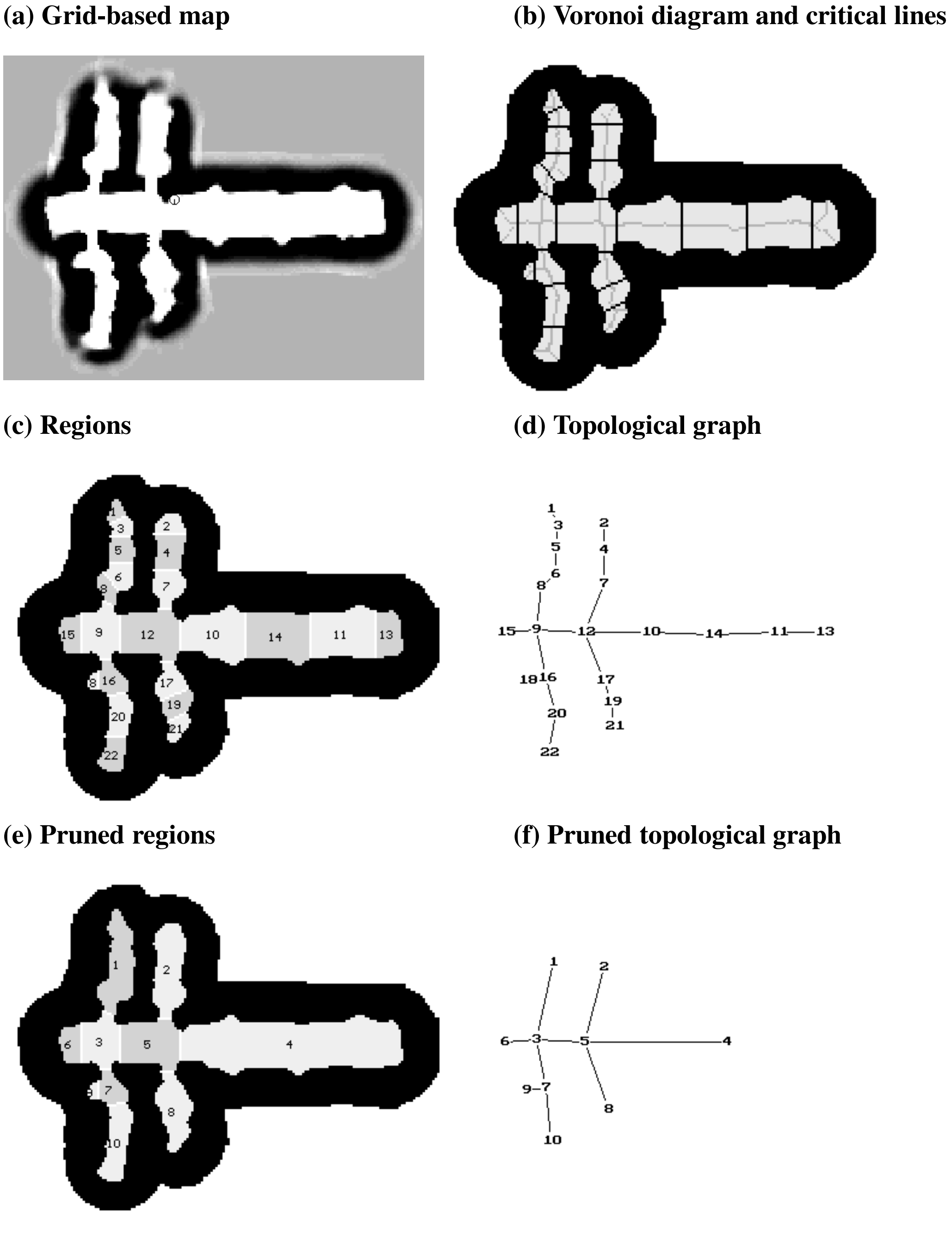


Fig. 15. Another example of an integrated grid-based, topological map.

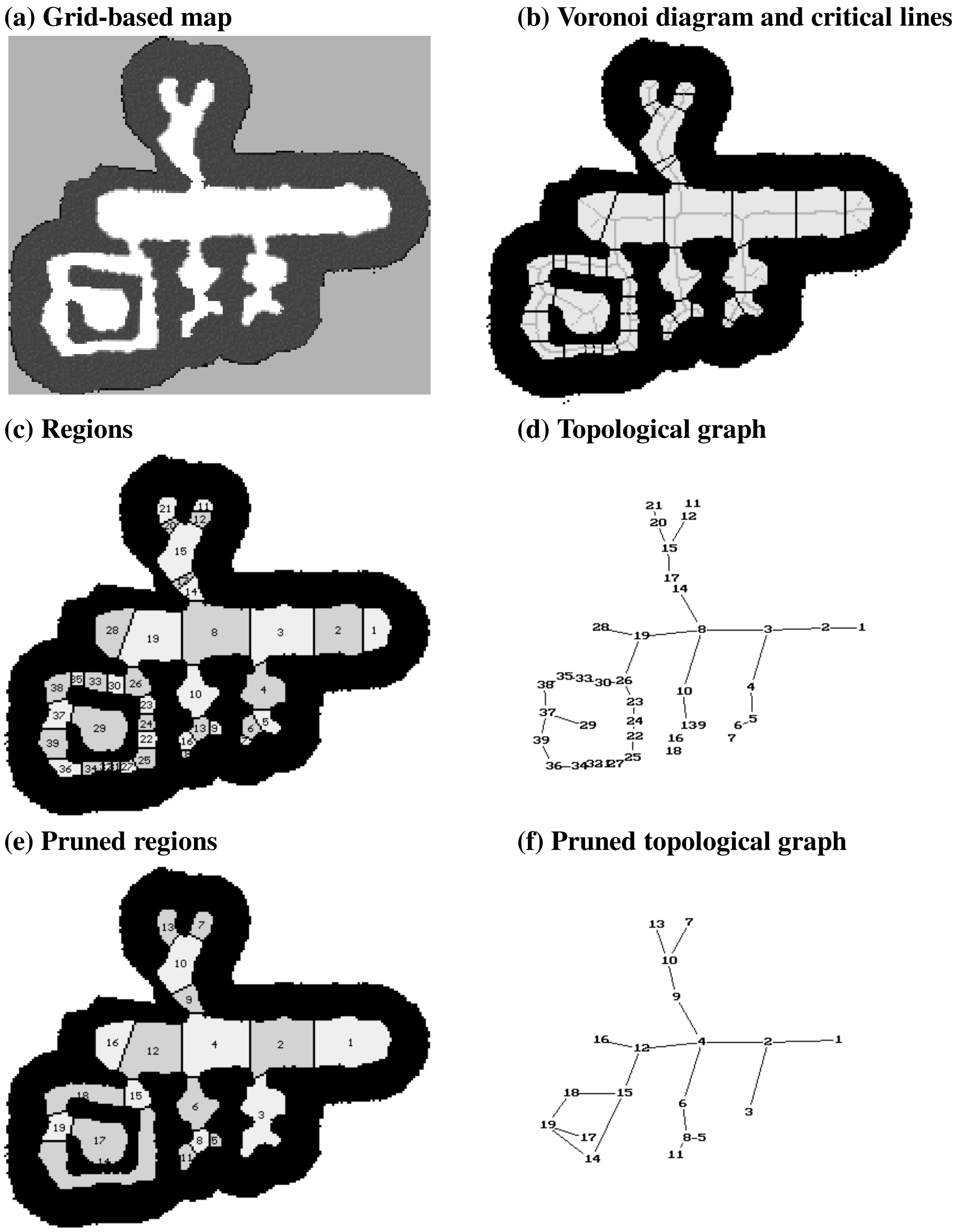


Fig. 16. A third example.

The compactness of topological maps—when compared to the underlying grid-based map—facilitates efficient planning. To replace the grid-based planner by a topological planner, the planning problem is split into three sub-problems, all of which can be tackled separately and very efficiently.

(i) **Topological planning.** First, paths are planned using the abstract, topological map. Shortest paths in the topological maps can easily be found using one of the standard graph search algorithms, such as Dijkstra's or Floyd/Warshal's shortest path algorithm [6], A* [75], or dynamic programming. In our implementation, we used the value iteration approach described in Section 2.5.

(ii) **Triplet planning.** To translate topological plans into motion commands, a so-called "triplet planner" generates (metric) paths for each set of three adjacent topological regions in the topological plan. More specifically, let $T_1, T_2, \ldots, T_n$ denote the plan generated by the topological planner, where each T_i corresponds to a region in the map. Then, for each triplet $\langle T_i, T_{i+1}, T_{i+2} \rangle$ ($i = 1, \ldots, n{-}1$ and $T_{n+1} := T_n$), and each grid cell in T_i, the triplet planner generates shortest paths to the cost-nearest point in T_{i+2} in the grid-based map, under the constraint that the robot exclusively moves through T_i and T_{i+1}. For each triplet, all shortest paths can be generated in a single value iteration run: Each point in T_{i+2} is marked as a (potential) goal point (just like the unexplored points in Section 2.5), and value iteration is used to propagate costs through T_{i+1} to T_i just as described in Section 2.5. Triplet plans are used to "translate" the topological plan into concrete motion commands: When the robot is in T_i, it moves according to the triplet plan obtained for $\langle T_i, T_{i+1}, T_{i+2} \rangle$. When the robot crosses the boundary of two topological regions, the next triplet plan $\langle T_{i+1}, T_{i+2}, T_{i+3} \rangle$ is activated. Thus, the triplet planner can be used to move the robot to the region that contains the goal location. The reason for choosing triples instead of pairs of topological regions is that instead of moving to the nearest grid cell in T_{i+1}, the robot takes into account where to move once T_{i+1} is reached, and approaches T_{i+1} accordingly.

(iii) **Final goal planning.** The final step involves moving to the actual goal location, which again is done with value iteration. Notice that the computational cost for this final planning step does not depend on the size of the map. Instead, it depends on the size and the shape of the final topological region T_n, and the location of the goal.

The key advantage of this decomposition is that almost all computation can be done off-line, for all path planning problems. For example, the map shown in Figure 14, which is the most complex map investigated here, has 67 topological nodes. Thus, there are only $67 \times 66 = 4422$ topological plans. Topological plans are symmetric. Thus, only half of them must be memorized. The map also has approximately 200 triplets, for which all triplet plans are easily computed. Thus, by decomposing the planning in a topological planning problem and a triplet planning problem, and by pre-computing and memorizing all topological and triplet plans, path planning amounts to table-lookup.

However, it should be noted that the topological decomposition does not change the (worst-case) complexity of the planning problem, so that all one can hope for is a constant speed-up. Assuming that the number of topological regions grows linearly with the size of the grid-based map, and assuming that the size of each region does not depend on the size of the map, topological planning using value iteration is quadratic in the size of the environment (just like planning using the grid-based map). The computational complexity of computing all triplet plans is linear in the length of the path and hence in the size of the map), as is the computation of all final goal-plans. In fact, computing *all* triplet plans and *all* final goal plans is still linear in the size of the grid-based map, so that the topological planner is the only non-linear component in the approach proposed here. Although both—regular grid-based planning and topological planning—require in the worst case time quadratic in the size of the world, the fact that topological maps are orders of magnitude more compact leads to a relative difference of several orders of magnitude. This huge difference is important in practice.

4 Performance Results

Topological maps are abstract representations of metric maps. As is generally the case for abstract representations and abstract problem solving, there are three criteria for assessing the appropriateness of the abstraction: *consistency*, *loss*, and *efficiency*.

(i) **Consistency.** Two maps are consistent with each other if every solution (plan) in one of the maps can be represented as a solution in the other map.
(ii) **Loss.** The loss measures the loss in performance (path length), if paths are planned in the more abstract, topological map as opposed to the grid-based map.

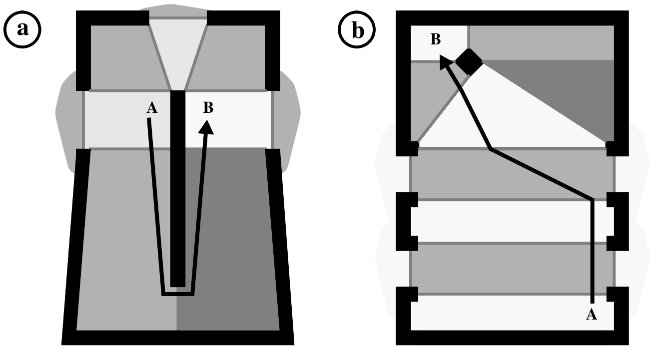


Fig. 17. Two examples in which the approach presented here yields suboptimal results. In both cases, the problem is to plan a path from “A” to “B”. (a) The topological planner will chose a sub-optimal path, since it leads only through two intermediate regions (as opposed to three). Such situations occur only if the topological graph contains cycles (which correspond to isolated obstacles in the thresholded grid-based map). (b) The triplet planner fails to move the robot on a straight line, since it looks only two topological regions ahead.

(iii) **Efficiency.** The efficiency measures the relative time complexity of problem solving (planning).

Typically, when using abstract models, efficiency is traded off with consistency and performance loss.

4.1 Consistency

The topological map is always consistent with the grid-based map. For every abstract plan generated using the topological map, there exists a corresponding plan in the grid-based map (in other words, the abstraction has the *downward solution property* [89]). Conversely, every path that can be found in the grid-based map has an abstract representation which is a admissible plan in the topological map (*upward solution property*). Notice that although consistency appears to be a trivial property of the topological maps, not every topological approach proposed in the literature generates maps that are consistent with their corresponding metric representation.

4.2 Loss

Table 2: Results (regular maps).

	map 1 (Figs. 9, 14)	map 2 (Figure 15)	map 3 (Figure 16)
grid cells	27,280	20,535	19,236
resolution	15 cm	10 cm	15 cm
cycles	8	0	1
topological regions	67	22	39
triplets	626	184	352
average shortest path length			
. . . meters (using grid-map)	15.87	9.42	11.55
. . . grid-cells	94.2	84.8	68.5
. . . topological regions	7.84	4.82	6.99
. . . meters (using topological map)	16.38	9.53	11.65
average loss			
. . . due to topological planning	1.82%	0.00%	0.03%
. . . due to triplet-planning	1.42%	1.19%	1.28%
. . . **total loss**	**3.24%**	**1.19%**	**1.31%**
total experiments	23,881,062	1,928,540	4,576,435
complexity			
. . . grid-based planning	$2.56 \cdot 10^6$	$1.74 \cdot 10^6$	$1.32 \cdot 10^6$
. . . topological planning	525	106	273
. . . **difference (factor)**	$\mathbf{4.89 \cdot 10^3}$	$\mathbf{1.64 \cdot 10^4}$	$\mathbf{4.83 \cdot 10^3}$

Table 3: Results (pruned maps).

	map 1 (Figs. 9, 14)	map 2 (Figure 15)	map 3 (Figure 16)
grid cells	27,280	20,535	19,236
resolution	15 cm	10 cm	15 cm
cycles	8	0	1
topological regions	40	10	19
triplets	222	30	166
average shortest path length			
. . . meters (using grid-map)	15.87	9.42	11.55
. . . grid-cells	94.2	84.8	68.5
. . . topological regions	6.12	3.25	4.65
. . . meters (using topological map)	16.51	9.45	12.20
average loss			
. . . due to topological planning	3.11%	0.00%	0.83%
. . . due to triplet-planning	0.94%	0.37%	5.22%
. . . **total loss**	**4.05%**	**0.37%**	**6.05%**
total experiments	23,881,062	1,928,540	4,576,435
complexity			
. . . grid-based planning	$2.56 \cdot 10^6$	$1.74 \cdot 10^6$	$1.32 \cdot 10^6$
. . . topological planning	245	32.5	88.4
. . . **difference (factor)**	$\mathbf{1.05 \cdot 10^4}$	$\mathbf{5.36 \cdot 10^4}$	$\mathbf{1.49 \cdot 10^4}$

Abstract representations lack detail. Thus, paths generated from topological maps may not be as short as paths found using the metric representation. For example, Figure 17a shows a situation in which a topological planner would chose a detour, basically because of the different sizes and shapes of the topological regions. Figure 17b depicts a situation in which the triplet-planner would give non-optimal results, since it determines the motion direction based on a limited look-ahead.

To measure the average performance loss, we empirically compared shortest paths found in a metric map with those generated using the corresponding topological approach, for each of the three maps shown in Figures 9, 14, 15, and 16. The results are summarized in Table 2. For example, for the map shown in Figures 9 and 14d, we conducted a total of 23,881,062 experiments, each using a different starting and goal position that were generated systematically with an evenly-spaced grid. The results are intriguing. Planning with the topological map increases the length of the paths by an average of 3.24%. In other words, the average length of a shortest path is 15.87 meters, which increases on average by 0.51 meters if robot motion is planned using the topological map. 0.28 meters (1.82%) are due to suboptimal choices by the topological planner, and the remaining 0.23 meters (1.42%) are due to suboptimal action choices made by the triplet planner. It is remarkable that in 83.4% of all experiments, the topological planner returns a loss-free plan. The largest loss that we found in our experiments was 11.98 meters, which was observed in 6 of the 23,881,062 experiments. Such loss was observed when the topological planner falsely assumed that the shortest route led through the large "foyer" (region 7 and 18 in Figure 14c). For example, when moving from region 58 to region 4 in Figure 14c, the topological plan ($\langle 58, 50, 39, 31, 18, 7, 2, 8, 3, 4\rangle$) leads through the foyer, which is clearly a detour.

Figure 18a shows the average loss as a function of the length of the shortest path. As can be seen there, for shorter paths the loss is a monotonically increasing function of the path length. As the path length exceeds 22.5 meters, the loss decreases. We attribute the latter observation to the fact that these paths are among the longest possible paths given the size of the environment, thus even a topological planner cannot increase the length of these paths any further.

The empirical loss for the maps shown in Figure 15 and 16 is even smaller, partially because there are fewer cycles in those maps. As summarized in Table 2, the average loss for the map depicted in Figure 15 is 1.19%, and the average loss for the map shown in Figure 16 is 1.31%. Figures 19a and 20a depict the loss as a function of optimal path length. Notice because there are no cycles in the second map (Figure 15), the topological planner always produces the optimal plan (i.e., a plan

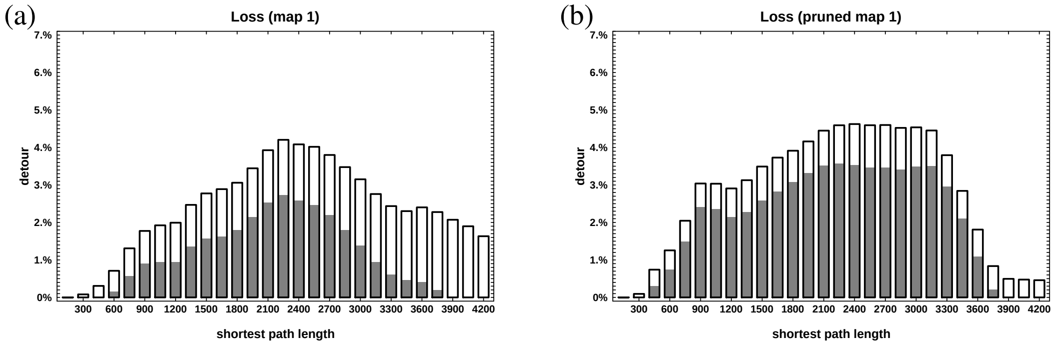


Fig. 18. Loss for paths generated for the map shown in Figures 9 and 14, using (a) the regular and (b) the pruned topological map. They grey portion of the loss is due to suboptimal action choices by the topological planner, while the white portion is due to the triplet representation.

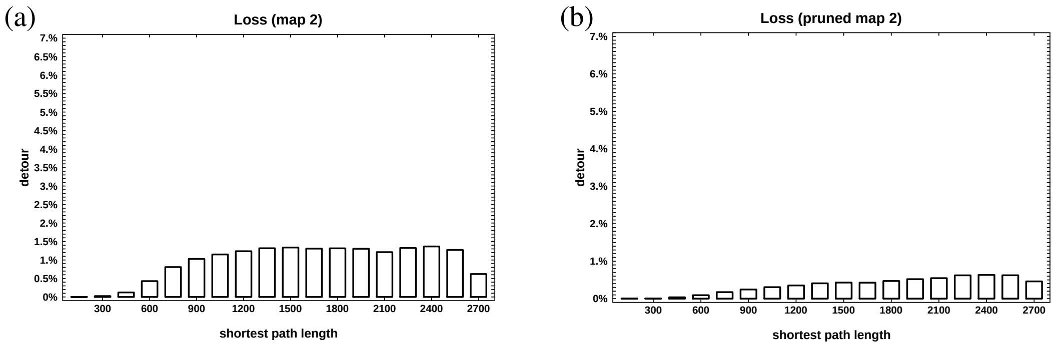


Fig. 19. Paths generated for the map shown in Figure 15, using (a) the regular and (b) the pruned topological map.

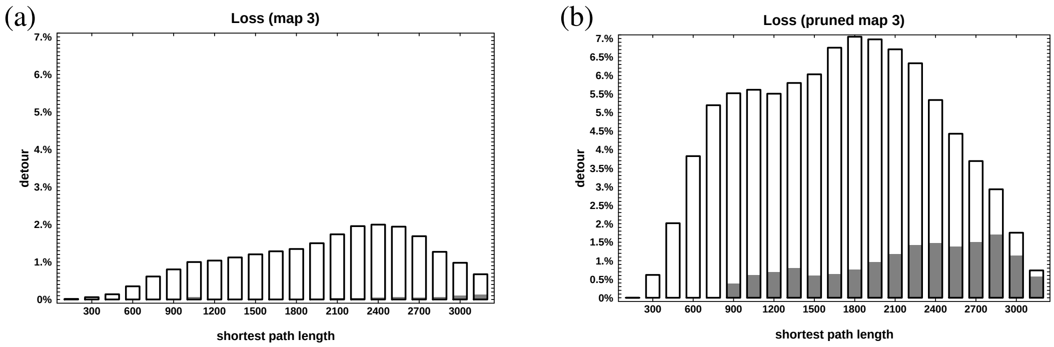


Fig. 20. Paths generated for the map shown in Figure 16, using (a) the regular and (b) the pruned topological map.

---

that includes the shortest path). Consequently, the 1.19% loss can be exclusively attributed to suboptimal action choices by the triplet planner. The 1.31% loss for the map shown in Figure 16 is mostly due to the triplet planner (1.23%), although rare topological detours infer an additional loss of 0.03%. Graphs illustrating the relative loss as a function of shortest path length are shown in Figures 19a and 20a.

We also investigated even more compact representations, such as those shown in Figures 14e, 14f, 15e, 15f, 16e, and 16f. These maps were obtained by *pruning* the original topological map: Pairs of adjacent regions are combined into a single region, if neither of them has more than two neighbors. Pruning subsumes series of nodes in long corridors into a single node (such as nodes 4, 13, 21, and 24 in Figures 14c and 14d), and also eliminates certain end-nodes (such as the region 17, 56, and 66, in Figures 14c and 14d). The results of experiments measuring the loss for these pruned maps are summarized in Table 3. For example, in 23,881,062 experiments using the pruned graph depicted in Figures 14e 14f, the average loss was 0.64 meters (4.05%), which is 26.1% larger than the loss inferred by the unpruned graph. For the map shown in Figures 15e and 15f, pruning actually *reduced* the overall loss to 0.37% (0.03 meters), which is only 31.0% of the loss inferred by the unpruned map. Finally, the pruned map shown in Figures 16e and 16f produces an average detour of 5.18% (0.70 meter), which is significantly larger (361%) than the loss inferred by the unpruned map—this difference is due to the fact that a long corridor is pruned into a single topological entity in Figures 16e and 16f. Figures 18b, 19b, and 20b depict the loss for the pruned map as a function of optimal path length. The shape of the curves here are similar to those obtained for the unpruned maps. Figure 19 illustrates once again that pruning reduces the loss for the cycle-free second map.

We conclude that the pruned graph is generally more compact. On the one hand, pruning can decrease the loss of the triplet planner due to the increased size of the topological regions, which typically yields improved triplet plans. On the other hand, the smaller number of regions in pruned maps usually induces additional loss on the topological planning level, if (and only if) the environment contains cycles. If the environment is cycle-free, the topological plans are identical for pruned and unpruned maps, since there exist only a single topological plan between each pair of points. Empirically, the increased loss on the topological level when pruning a map was found to outweigh the reduction of loss on the topological level. Pruning was only found to reduce the overall loss when the map is free of cycles.

4.3 Efficiency

The most important advantage of topological planning lies in its efficiency. Value iteration is quadratic in the number of grid cells. For example, the map shown in

Figure 9 happens to possess 27,280 explored cells. In the average case, the number of iterations of value iteration is roughly equivalent to the length of the shortest path, which in our example map is 94.2 cells. Thus, in this example map, value iteration requires on average $2.57 \cdot 10^6$ backups. Planning using the topological representation is several orders of magnitudes more efficient. The average topological path length is 7.84. Since the topological graph shown in Figure 14d has 67 nodes, topological planning requires on average 525 backups. Notice the enormous gain in efficiency! Planning using the metric map is $4.89 \cdot 10^3$ more expensive than planning with the topological map. In other words, planning on the topological level increases the efficiency by more than three orders of magnitude, while inducing a performance loss of only 3.24%.

The computational reduction is even more dramatic for the pruned maps, such as the one shown in Figures 14e 14f. This map consists of 40 nodes, and the average topological path length is 6.12. Consequently, topological planning is $1.05 \cdot 10^4$ more efficient than planning with the metric map, which is more than twice as efficient as planning with the unpruned map. However, as can be seen by comparing the results shown in Tables 2 and 3, the performance loss induced by the pruned map is 25% larger than the loss inferred by the unpruned map.

The map shown in Figure 15, which is smaller than the other maps but was recoded with a higher resolution, consists of 20,535 explored grid cells and 22 topological regions (unpruned map), or 10 regions (pruned map). On average, paths in the grid-based map lead through 84.8 cells. The average length of a topological plan is 4.82 (unpruned map), or 3.25 (pruned map, averaged over 1,928,540 systematically generated path planning problems). Here the complexity reduction is even more significant than in the first example. Planning using the metric map is a factor of $1.64 \cdot 10^4$ more expensive than planning with the topological map when using the unpruned map. This factor increases to $5.36 \cdot 10^4$ when using the pruned map. Clearly, since the pruned map exhibits a smaller loss, it is superior to the unpruned version in both categories: loss and efficiency.

Similar results are obtained for the map depicted in Figure 16. Here the planning complexity is reduced by a factor of $4.83 \cdot 10^3$ (unpruned map), or $1.49 \cdot 10^4$ (pruned map). While these numbers are empirical and only correct for the particular maps investigated here, we conjecture that the relative quotient is roughly correct for other maps as well.

It should be noted that in our implementation, every topological plan is pre-computed and memorized in a look-up table. Our most complex example maps contain 67 nodes, hence there are only 2,211 different plans that are easily generated and memorized. If a new path planning problem arrives, topological planning amounts to looking up the correct plan.

5 Related Work

The current approach draws on existing work on various aspects of AI and robotics. This section reviews related approaches in the areas of (1) map learning, (2) localization, (3) motion planning, (4) abstraction, and (5) learning automata. It points out various differences/commonalities to the work presented here and summarizes its main contributions.

5.1 Mapping

The key contribution of this paper is a new map learning method that integrates metric and topological representations. The vast majority of successful approaches to learning maps from sensor data focuses on a single type map, metric or topological, where topological maps are sometimes enriched by local metric information.

– **Metric approaches.** Occupancy grids, which form the basis of the metric component of our approach, are probably the most successful metric approach to mobile robot map acquisition to date. Occupancy grids have originally been proposed by Elfes and Moravec [31,32,70] and since been adopted in numerous robotic systems (e.g., [8,9,40,91,113]). Our approach differs from previous ones in that neural networks are used to learn the mapping from sensors to occupancy values; as a result, sensor readings are interpreted in the context of their neighbors, which increases the accuracy of the resulting maps [99]. Occupancy grids, however, are not the only metric representation. Chatila/Laumond [15] proposed to represent objects by polyhedra in a global coordinate frame. Cox [23] proposed to constructs probabilistic trees to represent different, alternative models of the environment. In his work, Kalman filters and Bayesian methods are used for handling uncertainty. Jeeves [101], an award-winning robot at the 1996 AAAI mobile robot competition [54], constructs geometric maps incrementally by concatenating wall segments detected in temporal sequences of sonar measurements. Jeeves's design was strongly inspired by the work presented here; its inability to handle dynamic environments and its strong commitment to parallel/orthogonal walls make its software approach significantly more brittle than the current approach.

– **Topological approaches.** Topological approaches represent maps as topological graphs, where nodes correspond to places and arcs correspond to actions for moving from one place to another. Often, topological graphs are enriched by local metric information to facilitate the navigation from one place to another.

  Among the earliest successful work in this field is an approach by Kuipers and Byun [57,58]. In their approach, topological places are defined as points that maximize the number of equidistant obstacles (a similar idea can be found in Choset's work, who refers to such as points as "meetpoints" [16–18]). Topological places are connected by arcs, which contain metric information for locally moving from one place to another. The approach disambiguates different places by local sensor information (taken at a single node or, if necessary, at a small number of neighboring nodes). In systematic simulations, this approach has been found to reliably learn large maps of indoor environments, even if sensor data is noisy. However, in these experiments the robot was equipped with a compass, which simplifies the localization problem significantly.

  A similar approach was proposed by Matarić [65]. Her algorithm acquires topological maps of the environment in which nodes correspond to pre-defined landmarks such as straight wall segments. Neighboring topological entities are connected by links. The topological representation is enriched by distance information to help keeping track of the location of the robot. The approach was evaluated on a physical robot and was found to be robust in practice. Its inability to maintain an exact position estimate imposes intrinsic scaling limitations. Moreover, since the recognition of landmarks in this approach involve robot motion, the approach might have severe difficulties in recognizing previously visited locations when approaching them from different directions (e.g., T-junctions).

  Another approach was proposed by Yamauchi and Beer's [112]. Their approach adds places into a topological graph whenever the robot's distance to previously defined places exceeds a certain threshold. Metric maps are used for localizing the robot (see discussion below). To determine the location of the robot within its map, the robot has to return close to its initial starting location (using only its wheel encoders), which imposes severe scaling limitations. Chown and colleagues [19] proposed a cognitively motivated approach to map learning, called PLAN. PLAN also learns a topological graph. Nodes in the topological graph are created whenever the robot enters a "choice point" (such as an intersection), or when new landmark comes into its field of view. At each of the nodes, the robot stores a collection of local views taken there. PLAN assumes that landmarks are uniquely identifiable, an assumption which simplifies the problem of place recognition (the correspondence problem, see below). In our work, this assumption is usually not fulfilled, due to the small amount of information conveyed by sonar measurements. Other topological approaches can be found in [73,105,115].

A key problem in map learning is to establish correspondence between current and past locations [7,22]. This problem is particularly difficult during map acquisition. In metric approaches, the correspondence problem is attacked exclusively through metric information; if the robot is capable of accurately estimating its coordinates in a Cartesian coordinate frame, the correspondence problem is solved. External sensor information is used to refine the metric position estimate. Topological approaches often take a different route. Correspondence between different places is usually determined based on a short history of sensor measurements. Some approaches, such as PLAN or the approaches in [59,73], require that different places are uniquely identifiable by the momentary sensor input. Other approaches, such as those described in [16–19,57,58,65,115], weaken this assumption by taking information from neighboring places and/or local metric information into account. To make sensor input easier to compare, many topological approaches require that the robot uses a specific navigation routine which ensures that the robot moves to specific points (such as meetpoints) before attempting to establish correspondence [57,58,65]. The accuracy required in metric approaches is usually higher than that required in topological approaches, since topological maps are more compact. There seems to be a tendency that topological approaches rely to a stronger degree on the robot's external sensors (such as cameras, sonars, compass or GPS), and to a lesser degree on the robot's odometry when compared to metric approaches. The willingness to ignore odometric information in topological approaches to map building becomes a severe scaling limitation when momentary sensor input is insufficient for the disambiguation of places. In situations such as the one shown in Figure 10, (odo)metric information is the key in establishing correspondence between current and past locations. Here topological approaches are typically at a disadvantage, since for robots equipped with sonar sensors most of the environment looks alike and correspondence cannot be established based on a short history of sensor measurements only. Metric approaches can cope with such situations much better, as best demonstrated by the empirical results described in this paper. They can also take momentary perceptual information such as landmark information into account, just like topological approaches.

The importance of integrating metric and topological maps for scaling up mobile robot operation has long been recognized. Among the first to propose this idea was Elfes [32] and Chatila/Laumond [15]. Elfes devised algorithms for detecting and labeling occupied regions in occupancy maps, using techniques from computer vision [32,31]. He also proposed building large-scale topological maps, but he did not devise an algorithm for doing so. Chatila and Laumond [15] proposed to represent objects by polyhedra in a global coordinate frame. From those they propose to decompose the free-space into a small number of cells that correspond to rooms, doors, corridors, and so on. While their paper contains most of the principle ideas, it unfortunately is in a proposal state where much of the algorithmic detail is missing. Neither of the above approaches has been shown to build maps that are significantly larger than the perceptual field of the robot, due to the difficulty of accurately determining a robot's position during mapping. We believe that our work is the first to fully implement these idea, and to get it to work for large-scale indoor environments. Because our approach integrates both representations, it gains advantages that were previously not available within a single approach, most notably: the ability to build large-scale maps even if sensor information is highly ambiguous and efficiency in planning.

5.2 Localization

Localization, that is, the problem of finding out where a robot is relative to previous locations and/or relative to its map, is one of the key problems in mobile robotics. A recent survey by Borenstein and his colleagues [7] dedicated exclusively to this topic illustrates the importance of localization and illustrates the large number of existing approaches. Cox [22] noted that "Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities"—an assessment that we do not share, but which nevertheless demonstrates the importance of the problem.

Algorithms for mobile robot localization can roughly be divided into two primary classes of approaches:

– **Landmark-Based Localization.** Landmark-based approaches use landmarks as references for determining a robot's position. It comprises by far the most popular family of approaches, partially because of its genuine computational simplicity, partially because landmarks appear to play a major role in human navigation [19]. Examples of successful algorithms for landmark-based localization can be found [5,21,52,47,50,74,76,80,95,111] and various chapters in [53].
– **Model matching.** Model matching algorithms extract geometric features from the sensor readings and match those to a model of the environment in order to identify errors in the robot's odometry [9–12,15,22,85,90,99,101,109]. Among the earliest work in this field is that of Moravec, Elfes, and Chatila/Laumond. Chatila and Laumond's approach [15] extracts geometric features such as line segments and polyhedral objects which are matched to a geometric map. Moravec

and Elfes, who pioneered the development of occupancy grids, were also the first to use occupancy grids for localization [31,71]. Just like the approach presented here, they proposed building local maps from single sonar scans and matching them to a previously learned (or hand-supplied) global map to identify errors in odometry. Their approach was recently re-implemented with minor modifications by Yamauchi and colleagues [112,113], who investigated its robustness to changes in the environment. Model matching can be computationally very expensive. This is because computing a single match requires many computations, prohibiting searching the space of all possible odometric errors exhaustively. It is common practice to search the space of odometric errors by hill-climbing [9,99,112,113].

Our approach falls into the second class: It is a version of model matching using metric maps. It differs from previous approaches in that the correspondence function is differentiable in the odometric error, which has two primary advantages: (1) The odometric error can be estimated with arbitrary (sub-grid cell) resolution. (2) Gradient descent is considerably faster. For example, Yamauchi and Langley [113] report that map matching using discrete hill climbing requires about 20 seconds (on a DECstation 3100). Our approach works in real-time (in the order of 0.3 sec on a 100Mhz Pentium computer), so that odometric errors can be corrected while the robot is in motion.

The vast majority of literature investigates mobile robot mapping and mobile robot localization separately. Interleaving mapping and localization is significantly more difficult than either task in isolation [85]. There are several attempts to integrate localization and mapping. For example, Leonard, Durrant-Whyte, and Cox [62] proposed a method that interleaves localization and mapping using Kalman filters [48] for position tracking. In their experiments, however, only the mapping component of their approach is demonstrated, leaving open the question as to whether these methods work well together in practice. Yamauchi and Beer [112] also interleave both localization and mapping. In their approach, the robot can only be localized at its starting location, forcing the robot to regularly return to its initial location. The approaches in [57,58,65] also interleave mapping and localization. They rely on landmarks to localize a robot, and also provide strategies for actively finding out if two places are the same if landmarks are ambiguous.

To the best of our knowledge, the maps presented here are the largest ever built autonomously using wide-angle sonar sensors and without a hardware mechanism for global positioning (such as a compass or GPS). The significance of this statement should be taken with a grain of salt, since different mapping approaches cannot be compared easily due to the different hardware and experimental conditions involved. Also, the reader should note that our approach rests on the orthogonal wall assumption, without which the approach might fail to map environments of the same size. To achieve the same robustness in environments that do not comply with this assumption, other, similar assumptions must be made.

The reader should note that the current approach is only able to localize the robot when its initial position is known. It is not able to localize a robot under global uncertainty, a problem which is also known as the "kidnaped robot problem" [33]. Only a small number of localization methods are capable of localizing a robot under global uncertainty, and all of those require (for obvious reasons) that the robot be equipped with a map of the environment [7,10–12,100].

5.3 Decomposition and Robot Motion Planning

The topological map extraction algorithm extracts a coarse-grained representation from high-resolution maps. Within the robot motion planning community, such algorithms are usually referred to as *cell decomposition methods* [92,60]. Within Artificial Intelligence, algorithms of this type are usually referred to as *abstraction algorithms* [43,51,89].

There is a huge body of literature on cell decomposition for robot motion planning. For example, Schwartz and Sharir published a series of five seminal papers in which the motion planning problem for various simple objects (such as ladders and disks) were solved in two- and three-dimensional spaces (see [92]). In several of these papers, the free-space is divided into a finite number of coherent regions, similar to the approach proposed in this paper. Once the free-space is partitioned, the robot motion planning problem can be solved by search of a (finite) graph. Similar cell decomposition methods and further references can also be found in Latombe's book on this topic [60], which provides an excellent survey on this topic. Most of these approaches assume that an accurate map of the environment is available prior to robot operation, in which obstacles are represented by polygons or circles. Most of the work on motion planning focuses on consistency (also called: completeness), that is, it seeks to establish algorithms which generate a solution if one exists, and returns a failure if no solution exists. Research on robot motion planning has also addressed issues of efficiency. A key difficulty arises from the observation that robot motion planning, in its general definition, is worst-case exponential in the number of degrees of freedom [13,84]. Due to the strong focus on consistency, O()-type complexity, worst case analysis and robots with many degrees of freedom, existing cell decomposition methods usually decompose the free-space in odd ways, which, if applied to mobile robot motion planning, do not at all maximize the run-time efficiency. Our method for extracting topological maps is specifically targeted at minimizing the performance loss for circular mobile robots. Therefore, boundaries of topological regions typically coincide with narrow regions such as doors, in which the robot is given little choice as to where to move. The triplet planner derives locally optimal plans, which have been designed to minimize the amount of loss suffered when planning topologically. We believe that the cell decomposition method proposed here yields more efficient control of a mobile robot than any of the other decomposition methods proposed in the robot motion planning literature. However, our method is currently restricted to the motion of a circular robot in a two-dimensional environment, whereas many of the methods listed above are applicable in higher-dimensional spaces.

Voronoi diagrams have previously been proposed for robot motion planning and mobile robot navigation [16,17,57,58,77]. These approaches use Voronoi diagrams as *road-maps* [14,60], i.e., they force a robot to move along the Voronoi diagrams. While traditional work on motion planning using Voronoi diagram rests on the assumption that an accurate model of the environment is available [60,77], Kuipers/Byun and Choset [16,18,57,58] have extended this framework to sensor-based motion planning. Their approaches enables robots to operate in the absence of a world model. Just like ours, their work assumes that the robot can sense the proximity of nearby obstacles (in Choset's approach the sensors must be noise-free). One of the striking advantages of Choset's work is that it can be applied in high-dimensional configuration spaces. However, forcing the robot to move along the Voronoi diagram yields suboptimal trajectories. The approach proposed here uses Voronoi diagrams for cell decomposition. To the best of our knowledge, this use of Voronoi diagrams is new, yet it has an obvious advantage of increased efficiency.

5.4 Abstraction and Dynamic Programming

As mentioned above, the topological maps described in this paper are a form of abstraction [89] and as such relate to the rich literature on abstraction in AI. The most closely related work on abstraction can be found in the literature on dynamic programming [4,44,83] and reinforcement learning [2,46,97]. In fact, our motion planning algorithm can be viewed as a model-based version of reinforcement learning [24,98,114]; however, for the sake of consistency with the literature we will refer to it as dynamic programming (there is no learning involved at the planning level).

In recent years, several researchers have proposed methods for solving dynamic programming problems by decomposing the state space into smaller subspaces. Dynamic programming is then applied hierarchically (1) to the subspaces and (2) on a more abstract level, where those subspaces are considered "abstract states." Existing approaches can roughly be divided into two classes, those that rely on a fixed decomposition, and those that decompose the state space by themselves during problem solving.

- **Fixed decomposition.** In [63,96,110] algorithms are presented that first learn solutions to sub-problems (using model-free reinforcement learning), then combine these solutions using a reinforcement learning algorithm. Sub-problems are specified through "sub-goals" or certain sub-reward functions, which have to be provided manually by the human designer.

  Dayan and Hinton [26] proposed a hierarchical reinforcement learning architecture which recursively decomposes the state space into squares of fixed size. At each level of control, policies are generated for moving from one square to a neighboring square. Their abstraction may be inconsistent. At higher levels of abstraction perceptual detail is omitted, which can turn a Markovian problem into a non-Markovian one for which no solution may exist.

  Dean and Lin [27] derived more general algorithms for solving dynamic programming algorithms efficiently given arbitrary partitions of the state space. One of their algorithms, called *Hierarchical Policy Construction method*, generates policies for transitioning from one region in the state space to another. On a more abstract level, those regions are considered "abstract states" just like in Dayan and Hinton's work, and dynamic programming is applied in this abstract (and potentially non-Markovian) state space. This paper goes beyond most of the work in the field in that it presents some useful formal convergence results for learning hierarchical control.
- **On-line decomposition.** Recently, Kaelbling [45] proposed an approach that decomposes the state space automatically based on a small set of randomly chosen "landmark states." In her approach, each landmark state defines a region and states other than landmark states are members of the region defined by the nearest landmark state. This approach is a version of Delaunay triangulation [29,39], a family of methods that decompose the state space through Voronoi diagrams. Just like in Dayan/Hinton's and Dean/Lin's approach, Kaelbling's approach applies dynamic programming at multiple levels: At the lower lever, local controllers are learned for moving from one region to another. On the higher level, an abstract control policy is learned which uses regions as abstract state descriptions. Her decomposition approach is similar to the one proposed here. The key difference lies in the way the state space is decomposed. By selecting

landmark states at random, the resulting decomposition is somewhat random. In contrast, our decomposition places topological transitions at narrow places of the environment. The practical implications of these different decompositions are formally not well understood. However, when applied to mobile robot navigation, we believe that the performance loss in our approach is smaller due to the more sensitive choice of the cross-region boundaries. A path planning approach for robots with excessive degrees of freedom that, similar to Kaelbling's approach, decomposes robot planning problems into sets of smaller problems by selecting a small number of random points can be found in [49].

Similar to Dayan and Hinton [26], Moore [68] recently proposed an approach for decomposing space into a set of rectangles, called *parti-game*. In his approach, the resolution of the decomposition is variable. It maximal along the boundary between obstacles and free-space. One of the nice properties of Moore's approach is its ability to deal with continuous spaces, just like most of the robot motion planning algorithms reviewed above. The parti-game algorithm does not take the actual path length into account during motion planning.

A method which attempts to find an *optimal* cell decomposition during problem solving is found in [104]. The SKILLS approach identifies regions in the state space (skills) in which the same policy can be used across different problems (tasks). This work makes a minimum of assumptions on the nature of the state space. For example, it differs from the work described in this paper in that it does not assume the availability of a model and in that it does not rely on a model of the environment or its geometry. Currently, its computational complexity limits its applicability to large state spaces. In fact, an initial attempt to use this method for generating a topological description of metric maps was less successful due to the enormous computational complexity involved. In principle, however, this approach can generate decompositions which might infer smaller losses than those described here.

Obviously, our approach falls into the second category, that is, it decomposes the state space automatically. The decomposition method proposed in this paper is specifically tailored towards mobile robot navigation, by placing the boundaries between adjacent topological regions at the narrow parts of the original state space. Neither of the existing approaches does this. With the exception of the SKILLS approach [104], none of the above-mentioned approaches takes optimality into account when selecting the boundaries between different regions.

Several of the aforementioned approaches [26,27,45,68] bear close similarity to the planning approach proposed in this paper: At the base level, dynamic programming is employed to generate plans for moving from a region to a neighboring region.

At a more abstract level, regions are treated as states, and dynamic programming is applied for finding global solutions to this (possibly non-Markovian) abstract model. Such hierarchical planning is very similar to the approach taken here in which a triplet planner solves local navigation problems and the topological planner generates global plans in the more abstract topological map. On the planning level, the only difference arises from the fact that the triplet planner considers three consecutive regions, whereas other approaches consider only two. By considering three adjacent regions, the performance loss is usually smaller, which comes at the expense of increased computational complexity.

5.5 *Learning Finite State Automata*

Within the AI community, research has been conducted on general methods that can reverse-engineer (learn) finite state automata based on their input-output behavior (see e.g., [3,20,78,66,72,86,87]). Finite state automata (FSAs) are learned by observing the result of sequences of actions. Often, algorithms capable of learning FSAs require a pre-given "homing sequence," i.e., a sequence that resets the state of the finite state machine (a routine that carries a robot to a unique location), or a sequence that produces observations that uniquely identify the resulting state. Some of these approaches require the FSA (the environment and robot's sensors) to be deterministic, whereas others can scope with certain types of stochasticity.

Approaches to learning FSAs differ from the approach taken here in (1) they make much fewer assumptions and hence can model a much larger variety of automata, and (2) they therefore scale poorly to environments of the size considered here. Scaling problems arise primarily because of three reasons:

(i) First, both the configuration space and the action space of a robot are continuous. Thus, the problem of map learning is not a problem of identifying a finite state machine. However, special-purpose navigation routines that safely carry the robot to a small number of geometrically distinguishable locations can make FSA learning algorithms applicable [3,58].

(ii) Second, work on learning finite state automata is usually not based in geometry. The work here assumes that the robot operates in a plane, for which basic geometric relations apply. For example, it is assumes that turning right 90° four times results in the same state. In the absence of such assumption, an approach that reconstructs the laws of motion by experimentation has to rediscover geometry.

(iii) Third, even if there were only finitely many states (e.g., as many as there are grid cells), in environments of the size described here, visiting every state is not practically feasible. In our approach, the robot uses its sensors to infer knowledge about states other than the current one. Work on the identification of finite state automata assumes that each state is visited at least once (in fact, states often have to be visited many times).

The reader should note that our approach is highly specialized to learning spatial maps, whereas methods for learning FSAs are targeted at different, more general problems of system identification [64]. Thus, while our approach is clearly better suited for learning maps, it lacks the generality of the FSA identification algorithms.

5.6 Contributions

The major contribution of the current paper is a working method for integrating metric and topological maps in map learning. Previous successful approaches to map learning with mobile robots were either metric or topological (sometimes enriched by metric information). While the idea of integrating metric and topological representations is not new [15,32,31], it has not yet been demonstrated that this can actually be done robustly in environments that are significantly larger than the perceptual range of a robot's sensors. Thus, the major contribution of the current paper is that it establishes a methodology for learning metric-topological maps, one which is demonstrated to work robustly in practice. The approach is demonstrated to inherit advantages from either paradigm: metric and topological. It inherits from metric maps the ability to robustly map large-scale environments, even if external sensor data is insufficient to establish correspondence between locations at different points in time. It inherits from topological maps the ability to plan orders of magnitude more efficiently, exploiting the fact that topological maps are more compact than grid-based maps.

On the technical side, the approach taken here draws strongly from the existing literature on mobile robot mapping, localization, motion planning, and abstraction. There are several new extensions to existing algorithms, the most significant of which are:

– Neural networks are used for interpreting the robot's sensors. The networks interpret sensor measurements in the context of neighboring measurements, which accounts for the robustness and the accuracy of the resulting occupancy maps.
– Voronoi diagrams are used for cell decomposition. The resulting decomposition places transition between different topological regions at narrow passages, which

is a good heuristic for minimizing performance loss.

– A hierarchical planner, consisting of a triplet planner, a topological planner and a final goal planner, reduces the complexity of motion planning. In environments of the size considered here, all plans (and all exceptions) can be entirely pre-computed. A modified dynamic programming method facilitates the efficient adaptation of plans when the model changes.
– A differentiable map matching function makes gradient descent applicable for efficient localization. The method has been demonstrated to successfully localize robots during mapping.
– An efficient dynamic programming has been developed which permits the efficient reuse of value functions even when the cost function changes.

All these advances are substantial for the current approach, which enables mobile robots to learn large-scale maps in real-time while moving at a speed of up to 90 cm/sec [36].

6 Discussion

This paper proposes an integrated approach to mapping indoor robot environments. It combines the two major existing paradigms: grid-based and topological. Grid-based maps are learned using artificial neural networks and Bayes's rule. Topological maps are generated by partitioning the grid-based map into critical regions.

Building occupancy maps is a fairly standard procedure, which has proven to yield robust maps at various research sites. To the best of our knowledge, the maps exhibited in this paper are significantly larger than maps constructed from sonar sensors by other researchers. Since neural networks interpret sonar readings in the context of adjacent sensor measurements, they do not assume conditional independence between adjacent sensor measurements—resulting in more accurate interpretations of sonar measurements. This paper also demonstrates that by integrating multiple sources of information, the robot position can be tracked accurately and in real-time in environments of moderate size—which is crucial for building metric maps.

The most important aspect of this research, however, is the way topological graphs are constructed. Previous approaches have constructed topological maps from scratch, memorizing only partial metric information along the way. This often led to problems of disambiguation (e.g., different places that look alike), and problems of establishing correspondence (e.g., different views of the same place). This paper advocates to integrate both, grid-based and topological maps. As a direct consequence, different places are naturally disambiguated, and nearby locations are detected as such. In the integrated approach, landmarks play only an indirect role, through the grid-based position estimation mechanisms. Integration of landmark information over multiple measurements at multiple locations is automatically done in a consistent way. Visual landmarks, which often come to bear in topological approaches, can certainly be incorporated into the current approach to further improve the accuracy of position estimation (see e.g., [55,100]). In fact, sonar sensors can be understood as landmark detectors that indirectly—through the grid-based map—help determine the actual position in the topological map (cf. [95]).

One of the key empirical results of this research concerns the cost-benefit analysis of topological representations. While grid-based maps yield more accurate control, planning with more abstract topological maps is several orders of magnitude more efficient. A large series of experiments showed that in a map of moderate size, the efficiency of planning can be increased by three to four orders of magnitude, while the loss in performance is negligible (e.g., 1.82%). We believe that the topological maps described here will enable us to control an autonomous robot in multiple floors in our university building—complex mission planning in environments of that size was intractable with our previous methods.

Despite these encouraging results, there is a variety of important open questions that warrant future research:

– **Sensor dynamics.** The current approach does not account for sensor drift or sensor failure. Once trained, the weights of the interpretation network are frozen. However, in principle it is possible to use a map as to generate targets for the interpretation network. As a result, the robot could constantly re-adjust its own interpretations. Empirically, we have found our approach to be surprisingly robust with respect to the failure of sensors.
– **Other sensors.** A second goal of future research is to incorporate other types of sensors; in particular, sensors that do not measure proximity. In an initial study, we extended the current approach by using a camera for floor segmentation and 24 infrared light sensors that measure proximity by measuring the intensity of reflected light [9]. The Bayesian approach to sensor integration described in this paper is flexible enough to accommodate other types of sensor information as well. In fact, in our initial experiments we found that the grid-based maps were more accurate when additional sensors were incorporated [103].
– **Dynamic environments.** While the current approach robustly handles dynamics in the environment (such as people, doors), it does not model them. It is an

open question as to how to incorporate models of moving objects into a grid-base representation. A recent study [91] has demonstrated that "semi-dynamic obstacles" (these are obstacles such as doors, whose presence might change but which are tight to a certain location) can be modeled by a variance analysis of grid-cell values. Further research is warranted to evaluate the robustness and utility of such approaches, and to model moving objects such as humans.

A key disadvantage of grid-based methods, which is inherited by the approach presented here, is the need for accurately determining the robot's position. Since the difficulty of position control increases with the size of the environment, one might be inclined to think that grid-based approaches generally scale poorly to large-scale environments (unless they are provided with an accurate map). Although this argument is convincing, we are optimistic concerning the scaling properties of the approach taken here. The largest cycle-free map that was generated with this approach was approximately 100 meters long; the largest single cycle measured approximately 60 by 20 meters. We are not aware of any purely topological approach to robot mapping that would have been demonstrated to be capable of producing consistent maps of comparable size. Moreover, by using more accurate sensors (such as laser range finders), and by re-estimating robot positions backwards in time (which would be mathematically straightforward, but is currently not implemented because of its enormous computational complexity), we believe that maps can be learned and maintained for environments that are an order of magnitude larger than those investigated here.

The approach described here has become part of a larger software package that is now distributed through one of the major mobile robot suppliers in the US (Real world Interface, Inc.) as the sole navigation software along with their B14 and B21 robots. It is already in use at more than 10 academic and industrial sites, where it has successfully mapped many different environments. An essential part of the software package is a fast, reactive collision avoidance routine, which is described elsewhere [36]. The advantage of integrating a fast collision avoidance routine is that dynamic obstacles and inaccuracies in the map do not lead to collisions. This module, combined with the mapping and planning approach described here, has found to navigate the robot reliably and with a speed of up to 90 cm/sec even in dynamic and cluttered environments. The University of Bonn's entry "RHINO" at the 1994 AAAI mobile robot competition, which won second price in the category "clean-up an office" and which was only defeated by a team of three collaborating robots [1], relied crucially on the mapping and exploration algorithms described in this paper (see [9,102,103]). Parts of the approach are also an integral part of a "museum tour-guide," a mobile robot which is currently being installed in cooperation with the University of Bonn at the *Deutsches Museum* in *Bonn*. The robot's task will be to interact with and to provide tours to visitors.

Acknowledgment

The author wishes to thank the RHINO mobile robot group at the University of Bonn, in particular Arno Bücken, Joachim Buhmann, Wolfram Burgard, Armin Cremers, Dieter Fox, Markus Giesenschlag, Thomas Hofmann, and Wolli Steiner for inspiring discussion and their contributions to the RHINO project. He specifically thanks Arno Bücken for re-implementing the techniques described in Section 3 and verifying the results. Some of the low-level software (TCX [35] and device drivers) were provided by the XAVIER mobile robot group at Carnegie Mellon University, which is gratefully acknowledged. The author thanks three anonymous reviewers for helping him improving the presentation of the material. He thanks Torsten Ihle and one anonymous reviewer for pointing out two errors in previous versions of this paper, and he also acknowledges the steady and helpful support by Real World Interface, Inc.

References

[1] T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. C. Santamaria. Io, ganymede and callisto – a multiagent robot janitorial team. *AAAI Magazine*, 16(1), 1995.

[2] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[3] K. Basye, T. Dean, and L.P. Kaelbling. Learning dynamics: system identification for perceptually challenged agents. *Artificial Intelligence*, 72:139–171, 1996.

[4] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[5] M. Betke and L. Gurvits. Mobile robot localization using landmarks. Technical Report SCR-94-TR-474, Siemens Corporate Research, Princeton, December 1993. will also appear in the IEEE Transactions on Robotics and Automation.

[6] J. Bondy and U. Murty. *Graph Theory with Applications*. Elsevier, New York, 1976.

[7] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.

[8] J. Borenstein and Koren. Y. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[9] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot Rhino. *AI Magazine*, 16(1), 1995.

[10] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, August 1996. AAAI, AAAI Press/MIT Press.

[11] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Position tracking with position probability grids. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996.

[12] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proceedings of IJCAI-97*. IJCAI, Inc., 1997. (to appear).

[13] J. Canny. *The Complexity of Robot Motion Planning*. The MIT Press, Cambridge, MA, 1987.

[14] J.F. Canny and B.R. Donald. Simplified voronoi diagrams. *Discrete and Computational Geometry*, 3:219–236, 1988.

[15] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985.

[16] H. Choset. *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, California Institute of Technology, 1996.

[17] H. Choset, I. Konuksven, and J.W. Burdick. Sensor Based Planning for a Planar Rod Robot. In *Proc. IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion on Multisensor Fusion and Integration for Intelligent Systems*, Washington, DC, 1996.

[18] H. Choset, I. Konuksven, and A. Rizzi. Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph. In *Submitted to Proc. IEEE Int. Advanced Robotics*, Washington, DC, 1996.

[19] E. Chown, S. Kaplan, and D. Kortenkamp. Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19:1–51, 1995.

[20] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinction approach. In *Proceedings of 1992 AAAI Conference*, Menlo Park, CA, July 1992. AAAI Press / The MIT Press.

[21] T.S. Collet and B.A. Cartwright. Landmark learning in bees. *Journal of Comparative Physiology*, January 1985.

[22] I.J. Cox. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[23] I.J. Cox. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–344, 1994.

[24] R.H. Crites and A.G. Barto. Improving elevator performance using reinforcement learning. In D. Touretzky, M. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press.

[25] J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 674–680, Scottsdale, AZ, May 1989.

[26] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

[27] T. Dean and S.-H. Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of IJCAI-95*, Montreal, Canada, August 1995. IJCAI, Inc.

[28] T. L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceeding of Seventh National Conference on Artificial Intelligence AAAI-92*, pages 49–54, Menlo Park, CA, 1988. AAAI, AAAI Press/The MIT Press.

[29] B.N. Delaunay. Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk (Bulletin of Academy of Sciences of the USSR)*, 7:793–800, 1934.

[30] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.

[31] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[32] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Department of Electical and Computer Engineering, Carnegie Mellon University, 1989.

[33] S. Engelson. *Passive Map Learning and Visual Place Recognition*. PhD thesis, Department of Computer Science, Yale University, 1994.

[34] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2555–2560, Nice, France, May 1992.

[35] C. Fedor. TCX. An interprocess communication system for building robotic architectures. programmer's guide to version 10.xx. Carnegie Mellon University, Pittsburgh, PA 15213, December 1993.

[36] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, to appear. (also appeared as Technical Report IAI-TR-95-13, University of Bonn, 1995).

[37] T. Fröhlinghaus and J.M. Buhmann. Real-time phase-based stereo for a mobile robot. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996.

[38] T. Fröhlinghaus and J.M. Buhmann. Regularizing phase-based stereo. In *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, 1996.

[39] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992. See also *17th Int. Coll. on Automata, Languages and Programming*, 1990, pp. 414–431.

[40] D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. Many robots make short work. *AI Magazine*, 18(1):55–64, 1997.

[41] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Pub. Co., Redwood City, California, 1991.

[42] R. Hinkel and T. Knieriemen. Environment perception with a laser radar in a fast moving robot. In *Proceedings of Symposium on Robot Control*, pages 68.1–68.7, Karlsruhe, Germany, October 1988.

[43] R. Holte. Speeding up problem-solving by abstraction: A graph-oriented approach. *Artificial Intelligence*, 1996.

[44] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press and Wiley, 1960.

[45] L. P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In P. E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173, San Mateo, CA, 1993. Morgan Kaufmann.

[46] L. P. Kaelbling, M. L. Littman, and A. W. Moore. An introduction to reinforcement learning. In L. Steels, editor, *The Biology and Technology of Intelligent Autonomous Agents*, pages 90–127, Berlin, Heidelberg, March 1995. Springer Publishers.

[47] L.P. Kaelbling, A.R. Cassandra, and J.A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[48] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[49] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics & Automation*, pages 2138–2145, San Diego, 1994.

[50] S. King and C. Weiman. Helpmate autonomous mobile robot navigation system. In *Proceedings of the SPIE Conference on Mobile Robots*, pages 190–198, Boston, MA, November 1990. Volume 2352.

[51] C.A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68:2, 1994.

[52] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[53] D. Kortenkamp, R.P. Bonassi, and R. Murphy, editors. *AI-based Mobile Robots: Case studies of successful robot systems*, Cambridge, MA, (to appear). MIT Press.

[54] D. Kortenkamp, I. Nourbakhsh, and D. Hinkle. The 1996 AAAI mobile robot competition and exhibition. *AI Magazine*, 18(1):25–32, 1997.

[55] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 979–984, Menlo Park, July 1994. AAAI, AAAI Press/MIT Press.

[56] R. Kuc and M.W. Siegel. Physically based simulation model for acoustic sensor robot navigation. *IEEE Transcation on PAMI*, 1987.

[57] B. Kuipers and Y.-T. Byun. A robust qualitative method for spatial learning in unknown environments. In *Proceeding of Eighth National Conference on Artificial Intelligence AAAI-88*, Menlo Park, Cambridge, 1988. AAAI Press / The MIT Press.

[58] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[59] B.J. Kuipers and T.S. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, Summer 1988.

[60] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[61] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, MA, 1992.

[62] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992.

[63] L.-J. Lin. *Self-supervised Learning by Reinforcement and Artificial Neural Networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.

[64] L. Ljung. *System Identification – Theory for the user*. Prentice-Hall, New Jersey 07632, 1987.

[65] M. J. Matarić. A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT, Cambridge, MA, January 1990. also available as MIT AI Lab Tech Report AITR-1228.

[66] R. A. McCallum. Instance-based state identification for reinforcement learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press. To appear.

[67] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[68] A. W. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 711–718, San Mateo, CA, 1994. Morgan Kaufmann.

[69] A.W. Moore and C.G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[70] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[71] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 116–121, 1985.

[72] M. C. Mozer and J. R. Bachrach. Discovering the structure of a reactive environment by exploration. Technical Report CU-CS-451-89, Dept. of Computer Science, University of Colorado, Boulder, November 1989.

[73] U. Nehmzow, T. Smithers, and J. Hallam. Location recognition in a mobile robot using self-organizing feature maps. In G. Schmidt, editor, *Information Processing in Autonomous Mobile Robots*. springer Verlag, 1991.

[74] H. Neven and G. Schöner. Dynamics parametrically controlled by image correlations organize robot navigation. *Biological Cybernetics*, 1995. to appear.

[75] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer Publisher, Berlin, New York, 1982.

[76] I. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, Summer 1995.

[77] C. O'Dúnlaing and C.K. Yap. A retraction method for planning the motion of a disk. *Journals of Algorithms*, 6:104–111, 1982.

[78] C.W. Omlin and Giles. C.L. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 45(6):937, 1996.

[79] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[80] L. Peters, H. Surmann, S. Guo, K. Beck, and J. Huser. Moria fuzzy logik gesteuertes, autonomes fahrzeug. 1994.

[81] D. Pierce and B. Kuipers. Learning to explore and build maps. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1264–1271, Menlo Park, July 1994. AAAI, AAAI Press/MIT Press.

[82] D. A. Pomerleau. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 19–43. Kluwer Academic Publishers, 1993.

[83] M.L. Puterman. *Markov Decision Processes*. John Wiley & Sons, New York, 1994.

[84] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proceesings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[85] W.D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2129–2197, Yokohama, Japan, July 1993.

[86] R. L. Rivest and R. E. Schapire. Diversity-based inference of finite automata. In *Proceedings of Foundations of Computer Science*, 1987.

[87] R. L. Rivest and R. E. Schapire. A new approach to unsupervised learning in deterministic environments. In P. Langley, editor, *Proceedings of the Fourth International Workshop on Machine Learning*, pages 364–375, Irvine, California, June 1987.

[88] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*. MIT Press, 1986.

[89] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[90] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1628–1634, San Diego, CA, May 1994.

[91] F. E. Schneider. Sensorinterpretation und Kartenerstellung für mobile Roboter. Master's thesis, Dept. of Computer Science III, University of Bonn, 53117 Bonn, December 1994. In German.

[92] J. T. Schwartz, M. Scharir, and J. Hopcroft. *Planning, Geometry and Complexity of Robot Motion*. Ablex Publishing Corporation, Norwood, NJ, 1987.

[93] D. Shepard. A two-dimensional interpolation function for irregularly spaced data. In *23rd National Conference ACM*, pages 517–523, 1968.

[94] R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(1), Spring 1995.

[95] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*, pages 1080–1087, Montreal, Canada, August 1995. IJCAI, Inc.

[96] S. P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 8, 1992.

[97] R. S. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Publishers, Boston, MA, 1992.

[98] G. J. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8, 1992.

[99] S. Thrun. Exploration and model building in mobile robot domains. In E. Ruspini, editor, *Proceedings of the ICNN-93*, pages 175–180, San Francisco, CA, March 1993. IEEE Neural Network Council.

[100] S. Thrun. A bayesian approach to landmark discovery and active perception for mobile robot navigation. Technical Report CMU-CS-96-122, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213, April 1996.

[101] S. Thrun. To know or not to know: On the utility of models in mobile robotics. *AI Magazine*, 18(1):47–54, 1997.

[102] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, August 1996. AAAI, AAAI Press/MIT Press.

[103] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, to appear.

[104] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press.

[105] M. C. Torrance. Natural communication with robots. Master's thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1994.

[106] S. Udupa. *Collision Detection and Avoidance in Computer Controlled Manipulators*. PhD thesis, Department of Mechanical Engineering, California Institute of Technology, 1977.

[107] V. Vapnik. *Estimations of dependences based on statistical data*. Springer Publisher, 1982.

[108] P. Wallossek. Realistische Simulation eines Mobilen Roboters in Echtzeit. Master's thesis, Dept. of Computer Science III, University of Bonn, 53117 Bonn, July 1995. In German.

[109] G. Weiß, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 595–601, 1994.

[110] S. Whitehead, J. Karlsson, and J. Tenenberg. Learning multiple goal behavior via task decomposition and dynamic policy merging. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 45–78. Kluwer Academic Publishers, 1993.

[111] E. Wolfart, R.B. Fisher, and A. Walker. Position refinement for a navigating robot using motion information based on honey bee strategies. In *Proceesings of the International Symposium on Robotic Systems (SIR 95)*, Pisa, Italy, 1995. also appeared as DAI Research Paper No 751, University of Edinburgh, Department of Artificial Intelligence, U.K.

[112] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Special Issue on Learning Autonomous Robots, 1996. also located at http://www.aic.nrl.navy.mil/∼yamauchi/.

[113] B. Yamauchi and P. Langley. Place recognition in dynamic environments. *Journal of Robotic Systems*, Special Issue on Mobile Robots, (to appear). also located at http://www.aic.nrl.navy.mil/∼yamauchi/.

[114] W. Zhang and T.G. Dietterich. High-performance job-shop scheduling with a time-delay td(lambda) network. In D. Touretzky, M. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1024–1030, Cambridge, MA, 1996. MIT Press.

[115] U.R. Zimmer. Robust world-modeling and navigation in a real world. *Neurocomputing*, 13(2–4), 1996.