Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT•

ELSEVIER

Cognitive Systems Research 4 (2003) 283–317

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies

Action editor: Risto Miikkulainen

Jean-Arcady Meyer[a,*], David Filliat[b]

[a]AnimatLab, Laboratoire d'Informatique de Paris 6 (LIP6), 8, rue du Capitaine Scott, 75015 Paris, France[1]
[b]DGA/Centre Technique d'Arcueil, 16 bis Av Prieur de la Cote d'Or, 94114 Arcueil Cedex, France

Received 29 June 2002; received in revised form 29 January 2003; accepted 24 February 2003

Abstract

This article reviews map-learning and path-planning strategies within the context of map-based navigation in mobile robots. Concerning map-learning, it distinguishes metric maps from topological maps and describes procedures that help maintain the coherency of these maps. Concerning path-planning, it distinguishes continuous from discretized spaces and describes procedures applicable when the execution of a plan fails. It insists on the need for an integrated conception of such procedures, which must be tightly tailored to the specific robot that is used, notably to the capacities and limitations of its sensory-motor equipment, and to the specific environment that is experienced. A hierarchy of navigation strategies is outlined in the discussion, together with the sort of adaptive capacities each affords to cope with unexpected obstacles or dangers encountered on an animat or robot's way to its goal.

Keywords: Autonomous mobile robot; Map-based navigation; Map-learning; Path-planning

1. Introduction

The capacity to navigate is obviously a major requirement for an animal striving to survive in a given environment, or for an autonomous robot trying to fulfill its mission, because it affords the possibility of finding energy sources while avoiding dangerous hazards. The most primitive navigation strategy that might help the animal or the robot succeed in such tasks consists of relying on mere chance and of moving randomly. As shown by Todd, Wilson, Somayaji, & Yanco (1994), evolutionary shaped blind action may suffice to keep a population of animats, i.e. of simulated animals or real robots (Meyer & Wilson, 1991; Meyer, Roitblat, & Wilson, 1993; Cliff, Husbands, Meyer, & Wilson, 1994; Maes, Mataric, Meyer, Pollack, & Wilson, 1996; Pfeifer, Blumberg, Meyer, & Wilson, 1998; Meyer, Berthoz, Floreano, Roitblat, & Wilson, 2000; Hallam, Floreano, Hallam, Hayes, & Meyer, 2002), alive and reproducing. However, the adaptive capacities of the animal or the robot would be dramatically enhanced if they used a slightly more elaborate navigation strategy, which would call upon some sensor to detect objects in the environment and on

*Corresponding author.
E-mail addresses: jean-arcady.meyer@lip6.fr (J.-A. Meyer), david.filliat@etca.fr (D. Filliat).
[1]URL: http://animatlab.lip6.fr

1389-0417/03/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1389-0417(03)00007-X

some reflex action to either pursue or avoid such objects. Additional adaptive capacities would be afforded to an animal or a robot able to turn to still more elaborate navigation strategies making it possible to elaborate and memorize a so-called *cognitive map* (Tolman, 1948), i.e. an internal representation of the environment. Survival in such case would depend not only on immediate perceptions and reflexes, but also on the recall of objects or events experienced in specific places in the past. Finally, because implementation of such internal representations would allow the animal or the robot to take into account that a given move from a given place would lead to another specific place, still more elaborate navigation strategies would afford planning capacities to the animal or the robot. In this case, survival depends on actions that are not only determined by current perceptions and memorized events, but also by their expected consequences.

A hierarchy of navigation strategies may accordingly be defined, depending upon how those combine past, present, and future information (Trullier, Wiener, Berthoz, & Meyer, 1997), the most elaborate relying on all three components at once. Incidentally, it also turns out that other behavioral capacities may be similarly classified (McFarland & Bösser, 1993; Mel, 1995).

This article reviews navigation systems implementing map-based strategies on mobile robots and discusses their respective adaptive capacities. It complements the review of biomimetic navigation systems of Trullier et al. (1997), which was essentially devoted to simulated animats. Being centered on map-learning and path-planning, it capitalizes on a review of localization strategies published in a companion paper (Filliat & Meyer, 2003), which are briefly summarized in the beginning of this article. Metric and topological map-learning systems are then successively described, along with path-planning systems applied to discrete and continuous spaces. The article ends with a discussion of the adaptive capacities of these implementations.

2. Map-based navigation

Basically, map-based navigation calls upon three processes (Levitt & Lawton, 1990; Balakrishnan, Bousquet, & Honavar, 1999):

- Map-learning, which is the process of memorizing the data acquired by the animat during exploration in a suitable representation.
- Localization, which is the process of deriving the current position of the animat within the map.
- Path-planning, which is the process of choosing a course of actions to reach a goal, given the current position.

Localization and map-learning are interdependent processes as using a map to localize a robot requires that the map exists, while building a map requires the position to be estimated relative to the partial map learned so far. In contrast, path-planning is a rather independent process that takes place once the map has been built and the animat's position estimated.

2.1. Available information

These three processes may rely on two distinct sources of information available to an animal, an animat or a robot. The first is the idiothetic source, which provides internal information about the animat's movements. This information may concern speed, acceleration, leg movement for animals, or wheel rotation for robots. Through straightforward integration, these data provide position estimates of the animat in a 2D metric space. This process is called dead-reckoning, path-integration or odometry.

The second source of information is the allothetic source, which provides external information about the environment. The corresponding cues may derive from vision, odor, or touch for animals, from laser range-finders, sonars or vision for robots, and they may be used in two different ways:

- they may be used to directly recognize a place or a situation. In this case, any cue, such as sonar time-of-flight, color or odor, may be used.
- they may be converted to information expressed in the 2D space related to the idiothetic data thanks to a metric model of the corresponding sensors. With such a metric model, it is possible to infer the allothetic cues that would be sensed in unvisited places, or to infer the relative positions of two places in which allothetic information has been gathered (Filliat & Meyer, 2003).

The drawbacks and advantages of these two sources of information are complementary. Indeed, the main issue raised by idiothetic information is that, because it involves an integration process, it is subject to *cumulative error*. This leads to a continuous decrease in quality, and therefore such information cannot be trusted over long periods of time. In contrast, the quality of allothetic information is stationary over time, but it suffers from the *perceptual aliasing* problem, i.e. the fact that, for a given sensory system, two distinct places in the environment may appear the same.

Consequently, in order to build reliable maps and to navigate for long periods of time, the two sources must be combined in a given animat (Cox, 1991). In other words, allothetic information must compensate for idiothetic information drift, while idiothetic information must allow perceptually aliased allothetic information to be disambiguated.

2.2. Map representation

When both allothetic and idiothetic sources of information are available, there are many ways to integrate them in a representation useful for animat navigation. Classically, the corresponding models are classified in two categories resorting to either metric or topological maps. In metric maps, the positions of some objects, mainly the obstacles that the animat may encounter, are stored in a common reference frame. In contrast, within topological maps, it is the allothetic characterizations of places that the animat can reach that are stored, along with some information about their relative positions (Fig. 1). Additional details about the advantages and drawbacks of these representations can be found in (Filliat & Meyer, 2003).

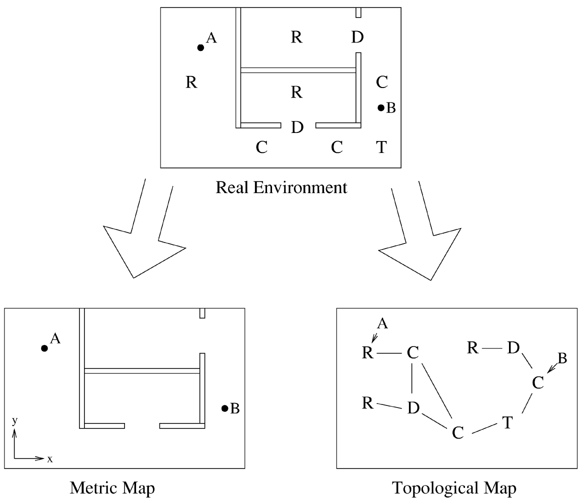

Fig. 1. Illustration of the classical distinction between metric and topological maps. In the metric framework, object positions are inferred and represented in a common reference frame. Two positions A and B are represented in this map by their coordinates in this reference frame. These coordinates make it possible to infer their distance. In topological maps, only place identities and their spatial relations are stored. Two positions A and B in the environment may be recognized as belonging to places R and C. This makes it possible to infer that position B can be reached from position A via places C, C and T (in this figure: C, corridor; D, door; R, room; and T, turn).

3. Map-learning

As previously mentioned, the map-learning process cannot be separated from the localization process. This interdependence makes map-learning difficult because errors in localization that arise during map-learning are incorporated into the map and subsequently need to be detected and corrected. The map-learning process, often referred to as SLAM, i.e. simultaneous localization and mapping, is a highly active area of research in the robotics community.

Furthermore, learning a map of an environment from scratch is intrinsically more difficult for an autonomous robot than localizing itself in a map that is already available. The main reason is that when the map is known, the robot is necessarily in some place represented in the map. Self-localizing then entails searching, through the possible positions, the place that best fits the available idiothetic and allothetic information. When the map has to be learned, an important additional problem is to evaluate whether the robot is in a part of the environment already stored in the map, or if it is in a part never visited before.

This evaluation step is crucial for obtaining a correct map and the specific issues it raises render useless a number of localization techniques that are not applicable when the map is only partially known. This is notably the case with almost every multiple-hypothesis tracking model, which rely on a complete map of the environment in order to estimate the relative credibility of the various hypotheses. If the real position is outside the mapped area of the environment, it cannot be estimated by these models. As the following paragraphs will show, these issues are usually tackled through the use of map-learning techniques that call upon single-position tracking algorithms for localization. This is possible thanks to the characterization of each position relatively to the previous one, independent of the fact whether it is located in the mapped area or not.

3.1. Metric map-learning

The simplest method for building a metric map calls upon an *incremental scheme* (Section 3.1.1) that consists of estimating the robot's position using the current map and in subsequently locally updating the map around this position whenever new information is acquired by the robot. The term incremental stems from the fact that the way new information is added to the map cannot be reconsidered afterward, even if this information turns out to be incorrect. For example, if the robot's position is wrongly estimated, any update of the map will be incorrect. If the robot later recovers its true position, it cannot take this information into account to cancel the effects of wrong past map updates. This incremental scheme may be problematic, for example, when large cycles, such as a corridor surrounding a room can be covered in the environment (Fig. 2). Upon closing such a cycle, a lot of information is gathered about the robot's previous positions but cannot be used, and the corresponding map-learning strategies often fail in such circumstances.

These limitations of the incremental scheme stem from the fact that, in general, the idiothetic and allothetic information gathered by the robot is memorized in a single common reference frame. No recording is made of what has been changed and of where the robot was when changes took place. Consequently, when a map update turns out to be erroneous, it is difficult to retrieve its effects and to reconsider them.

Using a map that records feature positions, it is possible to improve this incremental scheme through the memorization of all the confidences in interrelations between features (Section 3.1.2). This memorization, within the framework of a probabilistic scheme, allows new perceptual information to be propagated to unperceived features that are related to

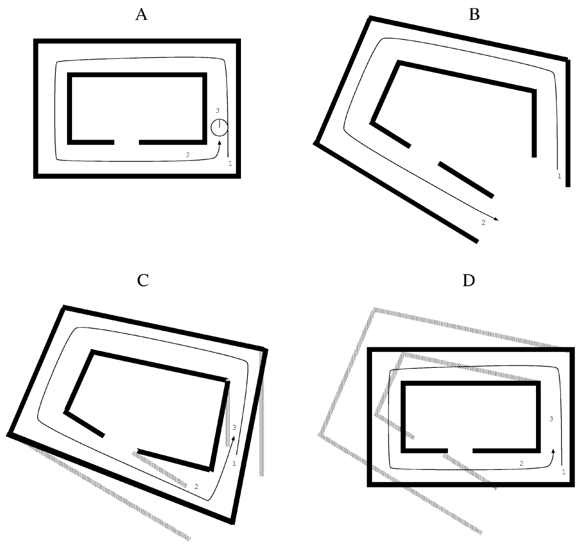

Fig. 2. (A) Real environment the robot is mapping and the trajectory taken by the robot. During mapping, errors accumulate from position 1 to position 2 and lead to a map where recorded positions do not correspond exactly to the real environment (B). Upon closing the cycle, at position 3, the robot can detect that its previous position estimates were wrong. In the incremental mapping framework, as previous map updates are not re-estimated, such errors are not fully taken into account and are only corrected locally (C). If all past map updates were reconsidered, it would be possible to recover the right map, by correcting all the previous position estimates (D).

perceived ones. This process affords possibilities of updating the global map, as well as of correcting past updates that turned out to be wrong in the light of new perceptions. In case of an environment containing a cyclic path, this process makes it possible to correct the map along the whole path, instead of merely correcting it around the position where the cycle is closed (Fig. 2D).

Another way to proceed with updating the global map is afforded by the decoupled recording of idiothetic and allothetic data so as to make the correction of past updates possible when new data about past positions become available (Section 3.1.3).

However, matching newly perceived data with data memorized in the map is a difficult problem, and using only the robot's currently estimated position to solve it can lead to poor performances. To improve these performances, particularly in cyclic environments, some models rely on more efficient data association methods, by resorting to off-line algorithms that allow all the available information to be efficiently taken into account (Section 3.1.4), or by resorting to on-line algorithms that approximate the off-line ones (Section 3.1.5).

It should be noted incidentally that all the models concerned in this section make use of a metric sensor model, which is mandatory for metric map-building (Filliat & Meyer, 2003).

3.1.1. Incremental mapping

In the context of *incremental mapping*, a position-tracking approach, which relies on the overlap between current allothetic data and data stored in the map, is usually used to estimate the robot's position with respect to the known part of the environment. These data may be memorized either as a set of features (such as segments, or corners) detected in the environment, or as an occupancy grid (see Filliat and Meyer, 2003 for details). In the first case, newly-perceived features are continuously added to the map. In contrast, in the second case, the environment is discretized into small cells and the probability of each cell being free or being occupied by an obstacle is updated in the light of new information.

In the context of *feature-based mapping*, Gasós and Martín (1997) advocate the use of fuzzy-segments to represent uncertainty in feature positions. Their model is based on the extraction of segments from sets of points provided by sonar sensors. Such segments are modeled by fuzzy-sets in the map's reference frame (Fig. 3). The size of a given fuzzy-set takes into account three sources of uncertainty: the scattering of the points around a segment, the distance of the points to the robot, and the uncertainty in the robot's position. Once fuzzy-sets have been created, colinear and overlapping fuzzy-sets are grouped in order to extract object's boundaries. Robot localization is performed by matching a local map, generated from recent perceptions, with the global map. To perform segment matching and merging, the use of fuzzy-sets is crucial in this approach because it makes it possible to quickly and efficiently detect the correspondence between perceived and stored segments.

Using an *occupancy-grid* framework (Moravec & Elfes, 1985), Thrun (1999) also implements an incremental mapping scheme. Within such a grid, the probability of each cell being occupied is updated using the Bayes rule, given the robot's position and the current sonar-sensor readings. This probability is computed using a neural network that has been trained by back-propagation in a known environment. The use of this neural network reduces the effects of spurious sensor readings. The robot's position is calculated using odometry and local map-matching, with a classical position-tracking approach. However, as this does not prove to be reliable enough for large environments, Thrun makes the additional hypothesis that walls are orthogonal. Such a hypothesis limits the estimation error in the robot's direction to values that permit local map-matching and efficient correction of the robot's position estimate. Thrun also resorts to an exploration scheme that allows the robot to drive towards unexplored areas, i.e. areas where cell probabilities have never been updated, so as to get a correct map of the whole environment as quickly as possible. For each cell, this scheme updates a value representing the distance to the closest unvisited cell area using a

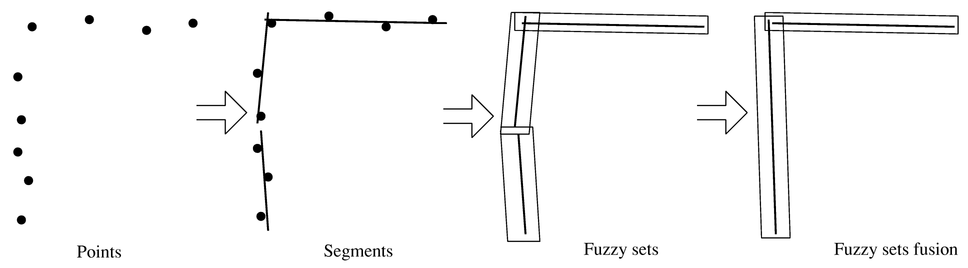

Fig. 3. Illustration of the use of fuzzy-sets for feature representation. Starting from points detected on obstacle surfaces, segments that match these points are extracted. Fuzzy-sets whose size depends on the quality of the approximation are then built around these segments. Finally, fuzzy-sets that are close and almost colinear are merged.

value-iteration algorithm (Sutton & Barto, 1998), so that performing a gradient descent on these values leads to unexplored areas.

Yamauchi, Schultz, & Adams (1999) provide a similar scheme but without using the orthogonal walls assumption. Their computation of occupancy probabilities is based on the combination of laser-scans and sonar-sensor values. This combination is designed to simultaneously avoid the use of spurious measurements from the sonar-sensors, and to filter too high laser-scan values due to the laser ray being targeted above the obstacles. Instead of using value-iteration, exploration is directed toward the closest *frontier* between explored and unexplored areas. The path to this frontier is computed using a depth-first search in known open-areas.

While Bayesian theory is widely used for representing and updating the likelihood of each cell being occupied, other techniques may be called upon. Hughes & Murphy (1992) describe the use of the Dempster-Shafer theory of *possibilities* (Dubois & Prade, 1986) for occupancy-grid mapping. This theory has the advantage of being able to represent partial information and of allowing the conflict between data to be assessed. The latter property makes it possible, for example, to detect in which region the map may be erroneous and should be refined. Whatever the case, results of occupancy grid mapping using the Dempster-Shafer theory closely resemble those obtained with probability theory (Murphy, 2000). It is also possible to use heuristic techniques, like the histogrammic in-motion mapping developed by Koren & Borenstein (1991a) to be used with sonar sensors. This method associates a score to each cell in the map, a high score meaning that the cell is occupied with greater certainty. A very simple metric sensor model is used, assuming that a single point in the sensor's direction is detected at the distance measured by the sensor. The score of the cell containing that point is simply increased, while the scores of the cells between the robot and this point are accordingly decreased by a smaller value. This method has the advantage of being computationally highly efficient, but may be difficult to adapt to a given robot and a given environment because of the numerous parameters that must be tuned (Murphy, 2000).

Arleo, Millán, & Floreano (1999) describe a variable-resolution map building method where each cell in the map is considered either occupied or unoccupied (Fig. 4). Starting from an initially empty world model, whenever a new obstacle is detected, the partition of the space around this obstacle is refined in order to incorporate an occupied cell which represents this obstacle. The obstacles are considered rectangular and their boundaries are detected using a line extraction algorithm on a local occupancy-grid map. Every time a new obstacle is detected, a contour-following strategy is used in order to model the whole obstacle. The odometry errors of the robot are compensated using the hypothesis that all obstacle boundaries are orthogonal, which makes the system very sensitive to the type of environment it is faced with. Exploration is directed toward cells that have been less often and less recently visited.

3.1.2. *Maintaining all features' inter-relations*

The main problem about incremental mapping is that it does not afford the possibility of recording

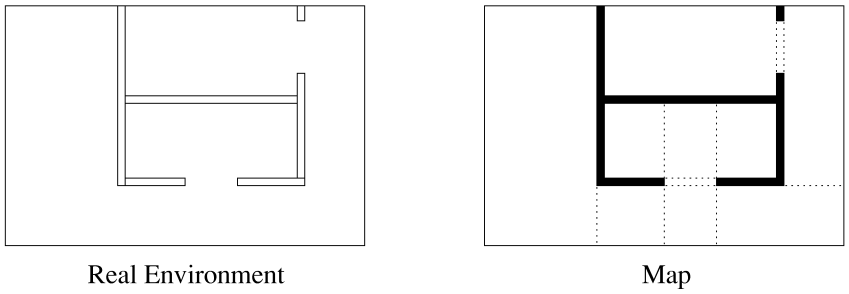

Fig. 4. Example of the discretization used by Arleo et al. (1999). The environment is discretized in cells whose boundaries are made of lines corresponding to obstacle boundaries. White cells represent empty space, while black cells correspond to obstacles.

dependencies between data added to the map. This, for example, precludes taking advantage of the fact that the relative positions of two landmarks are known very precisely, while their respective positions within the whole map are imprecise. When updating a map, it is important to take these dependencies into account so as to rely on accurate regions of the map in order to improve the inaccurate ones. This is possible thanks to a probabilistic scheme that maintains a *full posterior distribution* over the positions of the robot and the environmental features. This entails memorizing simultaneously the estimated positions, the variances of these estimates, and the covariances between these estimates. Note that this solution cannot be applied to the occupancy-grid framework.

This general approach is called *stochastic mapping* by Smith, Self, & Cheeseman (1988). In their framework, the positions of the robot and of all the features detected in the environment are stored in a vector that represents the environment. Additionally, a matrix is used to memorize the variances and the covariances of these positions. The corresponding model therefore assumes that these positions are uncertain and that their estimates are corrupted by a Gaussian noise. Covariances make it possible to record how the relative positions of two features were observed to change across successive detections performed by the robot. During map-learning, whenever a feature is perceived, its position in the map's reference frame is estimated from the robot's current position. If this feature has already been memorized in the map, Kalman filtering (Filliat & Meyer, 2003) is used to update the position and variance of the matched feature of the robot, and of all the other features, through their associated covariances. Unmatched features are simply added to the map at their perceived positions.

Various authors describe specific implementations of this approach. For example, the model of Ayache and Faugeras (1989) calls upon geometric features, such as lines extracted from a stereo-vision system. In this model, feature parameter uncertainties (like size and position) are represented by variances but, instead of calling upon covariances to represent relations between features, they use geometric relationships such as colinearity, parallelism or coplanarity. Such constraints, which are enforced through algebraic equations involving the feature parameters, make possible a significant reduction of the uncertainty attached to these parameters by propagating estimates along related features. Moutarlier & Chatila (1990) describe a similar model using segments representing boundaries of straight obstacles that are extracted as features from laser-scans. However, the authors remark that sequentially updating features perceived from a single location may lead to instability in the map building process. The reason is that the relative perceived feature positions when the robot stands still are less noisy than when the robot has moved between feature perceptions. Consequently, sequentially adding features integrates the robot's potential position error in the relative feature positions, and leads to biases in these position estimates. This problem is solved by estimating the robot's position using all the features perceived at a given place before updating these feature positions.

Still following the same scheme, Leonard, Durrant-Whyte, & Cox (1992) remark that uncertainty not only characterizes the position estimate of each feature, but also the very existence of each feature. This issue mainly stems from the use of sonar sensors for feature detection, which produces a lot of spurious measurements. To cope with such an issue, the authors associate to each feature a *credibility*, which is increased if a feature is reliably detected from several positions and which is decreased if a feature is not consistently detected. Consequently, this process makes it possible to remove from the map features that were erroneously detected, or features that have disappeared from the robot's environment because they were associated with a low credibility. Using only the most credible features to update the robot's position affords more robustness to the map-building process.

Feder, Leonard, & Smith (1999) improve the feature integration scheme by using a *delayed nearest neighbor* initiator instead of directly adding unmatched features to the map. The corresponding procedure filters out spurious measurements before adding features to the map. This entails delaying any insertion so as to check if the corresponding feature is consistently detected. The model also integrates an adaptive action selection mechanism, the goal of which is to direct the robot toward positions where the map's precision may be enhanced.

Leonard & Feder (1999) also remark that memorizing the covariances between all features and keeping them up to date, through a Kalman filter each time a feature is perceived, leads to a complexity in O(n^2) with the number of features, which is problematic for large environments. As simply ignoring covariances leads to instability (Hébert, Betgé-Brezetz, & Chatila, 1995), they resorted to a *decoupled stochastic mapping* procedure according to which the environment is represented by multiple regular rectangle sub-maps of limited sizes. These sub-maps are fixed a priori. Covariances are only maintained between features of each sub-map, thus limiting the complexity of the overall system. Experimental results show performances similar to those obtained with a model that maintains all the covariances.

The *Symmetries and Perturbations map* (SPmap) of Castellanos, Montiel, Neira, & Tardos (1999) is also a similar scheme that uses different methods for uncertainty representation. Uncertainties in feature positions are modeled probabilistically as in the previous models but, additionally, the theory of symmetries is used to represent the fact that the shape of a given feature limits the positional information that can be derived from its perception. For instance, matching two lines only provides positional information in the direction perpendicular to the lines, and not along their direction (Fig. 5). Knowing that the information provided by a given feature is partial then allows the feature and robot's position to be update selectively through Kalman filtering and affords a greater robustness.

An important practical issue raised by these approaches lies in the assumption that features should be correctly detected and identified, i.e. that the match between a memorized and a perceived feature is correct. This problem is often referred to as the *data association* problem. In most models, this association is simply based on the similarity between features and on their relative distances, assuming that a perceived feature matches a similar recorded feature if their distance is smaller than a given threshold. This approach makes feature identification simple, but it also makes it highly dependent on a correct initial estimate of the robot's position. For instance, Moutarlier & Chatila (1990) report that such matching is not reliable, because the positions of the features are evaluated relative to the robot's estimated position, which turns out to be very imprecise because of the robot's poor odometry. Their solution to this problem is to use a heuristic method to detect which perceived and recorded features have to be put in correspondence. Starting from the robot's estimated position, this heuristic finds the rotation and translation of the robot which lead to the best overall matching between perceived and recorded features. Using the corresponding displacement in the robot's position, feature associations are then sought that relate each feature to its

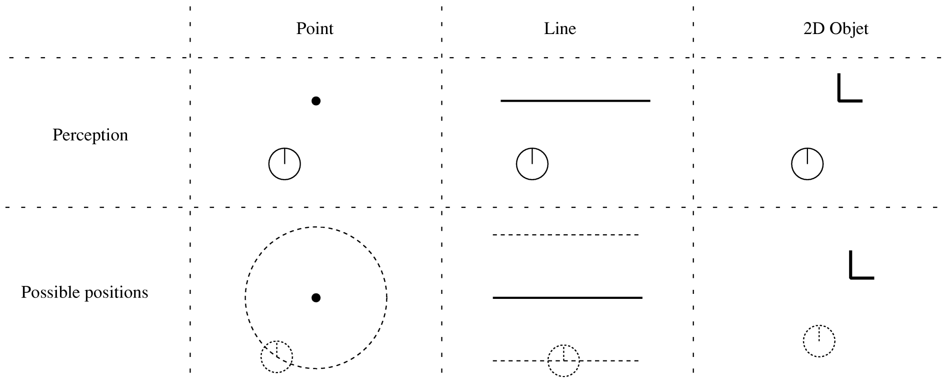

Fig. 5. Examples of positional information provided by the perception of various features. Perceiving a single point limits possible positions to a circle, perceiving a line limits them to two lines, while perceiving an object spatially extended in two dimensions corresponds to a single possible position.

closest neighbor. This makes it possible to reduce the effect of a poor estimate of the robot's position on feature matching.

3.1.3. Decoupling odometry and perceptions

The previous schemes integrate new information in the map in the form of feature positions estimated from sensor values and from the robot's positions. However, relative positions of objects estimated using sensor values acquired from a single robot position are usually much more precise than relative positions estimated while the robot is moving. The main reason for this fact is the difficulty of modeling some aspects of the odometry noise (e.g. sliding). This problem leads to several models in which the absolute position of the robot and the relative positions of the features are used in different ways in order to limit the effect of poor estimates in the robot's position. Moreover, separating these two sources of information is a further step toward the possibility of reconsidering past incorrect map updates, because it makes it possible to reconsider past erroneous data association when new information on past positions becomes available.

Extending the *stochastic mapping* scheme previously described, Hébert, Betgé-Brezetz, & Chatila (1996) suggest decoupling odometry and perceptions. To this end, the environment is represented by multiple sub-maps, each sub-map containing only positions and covariances of features whose mutual spatial relations have been observed from a single position (Fig. 6). Each sub-map has an associated position in a global map that is estimated using the robot's odometry between the points where the features of the sub-maps have been detected. When a new set of features is perceived, the possibility that any such feature is already included in one sub-map is checked. If this is the case, all the perceived features are added to the corresponding sub-map and the parameters of the features in the sub-map are updated using the relative positions of the newly perceived features. If no feature already belongs to an existing sub-map, a new sub-map is created that contains the new features. Whether the two sub-maps should be merged because the new set contains features belonging to both of them is also checked. In this model, the robot's odometry only influences sub-map positions in the global map. Once the map is updated, the robot's position is estimated using its relative position to perceived features via a Kalman filter. According to the authors, a more precise map is obtained with this scheme than with the standard *stochastic mapping*. Moreover, the complexity of the algorithm is reduced by computing feature covariances among features that belong to the same sub-map only.

Borghi and Brugali's extension of Engelson's *diktiometric map* (Engelson & McDermott, 1992; Borghi & Brugali, 1995) follows the same principle. The features that are exploited by their model are corners detected by a laser range-finder. The corresponding map is represented as a set of sub-maps similar to those of the previous model with relative positions estimated through odometry (Fig. 7). In this model, no global position estimate is computed, but the robot's position is monitored within the sub-map it is currently in. To achieve this, when a new set of features is perceived, this set is compared to all the sub-maps already stored. If some perceived features can be identified as belonging to an already

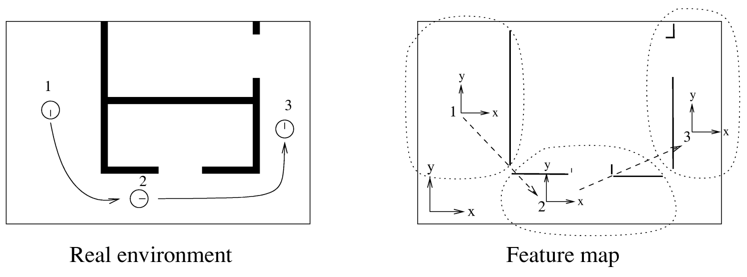

Fig. 6. Illustration of the map decomposition used by Hébert et al. (1996). Each sub-map (1, 2 and 3) groups features that have been perceived from the corresponding position in the environment.

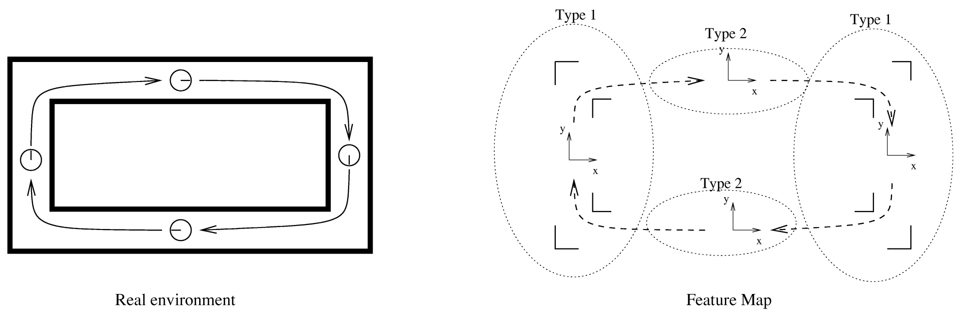

Fig. 7. Map representation used by Borghi & Brugali (1995). The environment is memorized as a set of sub-maps along with their relative positions. Two type of sub-maps are used. Type 1 corresponds to parts of the environment where features (corners in this case) can be perceived. Type 2 corresponds to parts of the environment that are featureless, such as corridors. The relative positions of the sub-maps are estimated through the robot's odometry.

stored map, the robot's position is computed inside this sub-map, using these features, and the corresponding features of the sub-map are updated. If none of the perceived features already exists in the map, a new sub-map is created with these features. In these two cases, the position of the sub-map is evaluated relative to the last known position through odometry.

3.1.4. Reconsidering past updates off-line

In all the previous models, associations between perceived and stored data are made once and are not reconsidered afterwards. If the robot's position turns out to be poorly estimated, for example if an explored part of the environment is encountered after closing a large cycle, past data associations and map updates cannot be modified to take this information into account. As reconsidering such past updates would lead to better maps, some models tackle this issue.

As suggested in the Introduction, the ideal solution would be to look forward in time and take into account at each time-step any future information about the robot's position. To achieve this, some models resort to off-line algorithms that work on data sets of idiothetic and allothetic information gathered by the robot. Indeed, these sets characterize the positions they record by both past and future information. This is for example the case with several models based on POMDP's, the localization methods of which were reviewed in Filliat & Meyer (2003). These models are based on an *expectation-maximization* (EM) algorithm (McLachlan & Krishnan, 1997) that iterates two stages referred to as the E-step and the M-step (Fig. 8). Given an approximate map, a sensor model, a movement model, and some training data recorded by the robot, the E-step evaluates the robot's most likely trajectory. This step is very similar to Markov localization (Filliat & Meyer, 2003), but the strength of this version lies in the fact that it takes all the information brought by the training data set into account, including future cues about the robot's position (Fig. 9). In the M-step, a map is computed that best reflects the

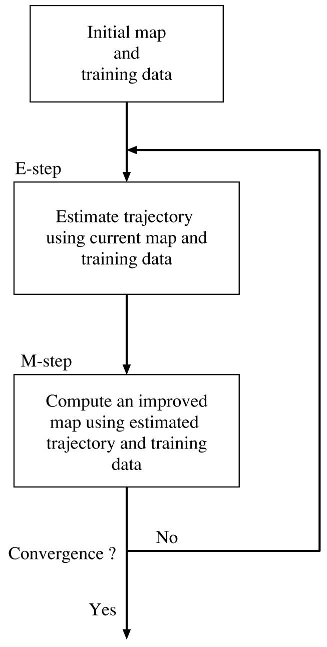
Initial map and training data
E-step
Estimate trajectory using current map and training data
M-step
Compute an improved map using estimated trajectory and training data
Convergence ?
No
Yes

Fig. 8. Illustration of the EM algorithm (see text for details).

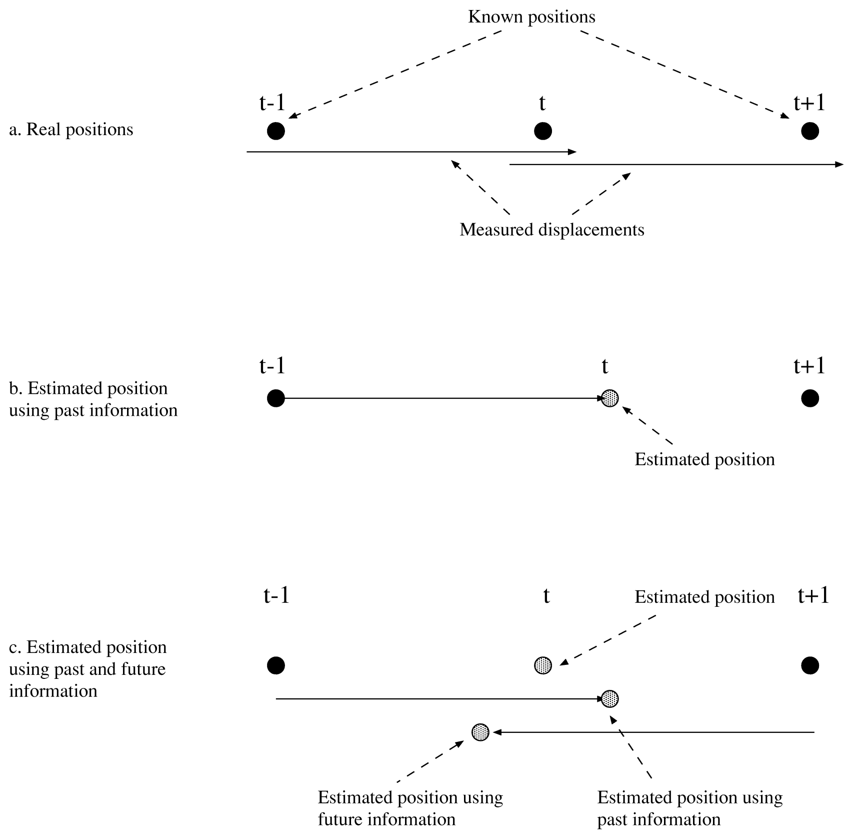
Known positions
t-1
t
t+1
a. Real positions
Measured displacements
b. Estimated position using past information
Estimated position
c. Estimated position using past and future information
Estimated position using future information
Estimated position using past information

Fig. 9. Illustration of the effect of taking future information about the robot's position into account. In this example, position at time t has to be estimated knowing positions at time $t-1$ and $t+1$. All the measures of displacements are over-estimated. Using future information allows the position to be more efficiently estimated.

gathered data along the trajectory calculated in the first stage. This map computation entails a data association different from those of the initial map which is more correct and leads to a better map. Iterating these two steps allows to converge toward a map that maximizes the likelihood of the gathered data. This algorithm, however, can get easily trapped in local minima, and careful attention has to be paid to the initial data which are used.

Thrun, Fox, & Burgard (1998) implement this algorithm using punctual landmarks detected by the robot and laser-scans. Two EM algorithms are successively used. In the first one, perceptions are the types and approximate positions of landmarks relative to the robot. Landmarks may be detected in a number of ways and do not need to be individually recognizable. Given a set of landmark perceptions and robot movements, the EM algorithm is used to compute a map containing the positions of all the landmarks and corresponding to an environment in which the robot could gather these data with a high probability. This algorithm is efficient, as the corresponding search space, which contains a few landmarks only, is significantly smaller than the search space of metric maps that would contain all obstacles detected by a laser range-finder. Once this landmark map has been computed, data about the robot's position that are contained in the training set are corrected using the now-known landmark positions. These corrected data, together with laser-scan data, are then used to initialize a second EM algorithm that computes a metric representation of the environment. However, in order to make this second algorithm computationally tractable, position probabilities have to be approximated by Gaussians. This approximation makes the algorithm very sensitive to initial conditions and often leads to a local optimum. But, in this context, as the landmark map has been used to correct most of the odometric error, such local optimum is probably a good approximation of the optimal metric map.

The implementation by Burgard, Fox, Jans, Matenar, & Thrun (1999) of this algorithm is based on a map that stores local occupancy grids along with their relative positions estimated via odometry. This approach is used to both overcome the local minima issues that arise in the previous models, and to avoid the use of a landmark map as an intermediate representation for occupancy grid mapping. To achieve this, the second step of the EM algorithm, which is only guaranteed to converge toward a local extremum in the general case, is modified to incorporate a simulated annealing variant. As a result, convergence towards a correct map may be achieved using noisier data.

3.1.5. Reconsidering past updates on-line

The models of the previous section allow precise maps to be computed using all available data but require resorting to off-line map-building. This off-line step often requires the intervention of a human operator and thus reduces the robot's autonomy. However, approximations of such models may be used, thus allowing most of their desirable properties to be kept, while making on-line processing of new data possible.

In the model of Lu & Milios (1997), the map is composed of a set of laser-range scans, along with their spatial relationships (Fig. 10). These relationships are of two kinds, the first one being derived from the robot's odometry between the points where the scans were taken, the second one being derived by scan-matching when scan positions are close enough for the corresponding measurements to overlap. An optimization process computes the position of each scan maximally satisfying spatial relationships, taking into account the uncertainty associated with each type of relation and giving greater confidence to relationships derived from scan-matching than to those derived from odometry. The result is a map exhibiting a minimal number of discrepancies and that best reflects the environmental layout. This mapping method allows past map updates to be reconsidered because, whenever a cycle is closed in the environment, a new relationship derived from scan matching is added to the map. This new relationship enforces the fact that the current robot position corresponds to the position at the beginning of the cycle. Using this new relationship to optimize the scan positions therefore entails modifying the positions of the scans along the whole cycle. As a consequence, past map updates are thus revised.

Gutmann & Konolige (2000), considering that the approach of Lu and Milios only works well with good initial position estimates, improve the corresponding scheme. When no cycle is closed in the environment, the technique of Lu and Milios is used, but applied to a few past scans only. This proves to be reliable enough to build an approximately correct map of the environment and leads to results similar to those of incremental mapping. A separate procedure is responsible for detecting cycles in the

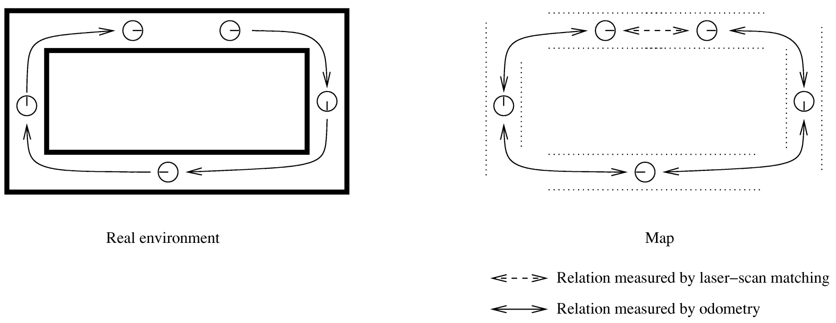
Real environment
Map
Relation measured by laser–scan matching
Relation measured by odometry

Fig. 10. Map representation used by Lu & Milios (1997). A set of laser-scans are memorized along with the relations between the positions from which they were taken. These relations may be derived either from the robot's odometry or from scan matching.

environment and reconsidering past updates in this case. This cycle detection is achieved by matching a local map composed of a few past scans with the global map in an area surrounding the robot's position estimate. The size of this area grows with position uncertainty. When a good enough match is found, a loop is considered to be closed, and the technique of Lu and Milios is used to optimize the positions of all the scans in the loop. This method provides faster and more accurate results.

Thrun, Burgard, & Fox (2000), using a set of laser-scans along with the position from which they were recorded as a map, describe a similar model. Their model uses Markov localization in order to estimate the robot's position. The mapping strategy is a weaker version of the expectation-maximization described above, as it cannot take into account any future information for localization. However, similar properties are achieved by using special procedures that modify past position estimates when a cycle is closed. In normal operation mode, whenever a new scan is acquired, it is simply added to the map at the robot's most likely position. This step is similar to incremental mapping, except that the use of Markov localization allows local minima to be avoided because it finds the current best position estimate for the robot. A different procedure is applied when a cycle is closed in the environment. The error in the robot's position computed when re-observing a part of the map is propagated to all the scans belonging to the loop, the positions of which are optimized so as to maximize the probability of the perceived data. The result is a close approximation of what would have been achieved using EM and taking the whole future information into account.

3.1.6. Topological map generation

As mentioned in the companion paper on localization (Filliat & Meyer, 2003), it may be useful to extract a topological representation of the environment from a metric map, mainly in order to facilitate planning tasks.

Some topological information may already be included in the metric map representation. For example, when odometry and perceptions are decoupled, as in the model of Borghi & Brugali (1995), although the global map is metric, it is nevertheless represented as places that are related via idiothetic cues. The model of Arleo et al. (1999) also contains topological information in the arrangement of cells that discretize the free space.

Other models explicitly extract topological information once the metric map has been obtained. Chatila & Laumond (1985) decompose the free-space between objects represented by polyhedra into cells that correspond to rooms and corridors. Thrun (1999) decomposes the free space represented by an occupancy-grid using a Voronoï diagram, the critical points of which are used to find the boundaries of the topological regions. As a result, regions are separated by narrow passages, such as doorways. Buhmann et al. (1995) also extract topological information from occupancy-grids. Their approach calls upon a database of small labelled occupancy-grids that represent doors, corridors and rooms of the environment. The maps of this database are continuously matched to the map of the environment and, when a convenient matching is found, the associated label is assigned to the corresponding region in the map.

Although such discretization procedures are very useful to reduce the complexity of subsequent path-planning, it should be noted that node identification relies on a prior estimate of the robot's metric position, which means that topological information is not used for localization in the corresponding models.

3.2. Topological map-learning

Topological map-learning is relatively different from metric map-learning, as place definitions and their relative positions are recorded, respectively, in the map nodes and links. Most of the time, nodes store allothetic place definitions without the need to resort to metric models of the sensors, while links memorize relative positions of the nodes, ranging from simple adjacency information to precise relative metric positions. This separation of the two types of information naturally makes it possible to reconsider past map updates more easily than in most metric map-learning models. However, node recognition, as a counterpart of data-association, is often a difficult task which can be highly sensitive to perceptual aliasing and perceptual variability.

Topological map-learning often entails recognizing if the current situation corresponds to a node in the map, and adding a new node if no such node is found (Section 3.2.1). Link information is then updated using idiothetic data. When this information is metric, the corresponding strategies usually require relatively precise idiothetic data. As raw data are often of poor quality, they have to be corrected, using either the existing map or external means. Section 3.2.2 describes these methods.

In many models, the information memorized in links is not required to be precise and consistent over the whole map (Fig. 11), i.e. it just needs to be locally consistent. As a consequence, taking new information about past positions into account is usually less important than in metric map-learning. Update strategies are therefore often local (Fig. 12C), as a counterpart of the incremental scheme

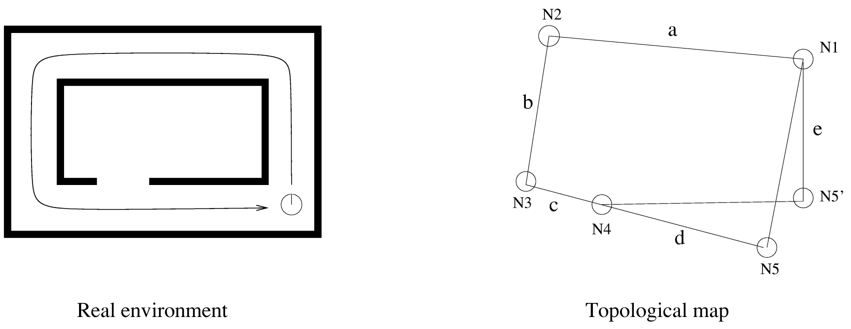

Fig. 11. A topological map, which memorizes metric relations in its links, may be inconsistent, i.e. two different paths between two nodes may record different relative positions for these two nodes. In this example, the position of N5 relative to N1 through the successive connections a, b, c and d is different from the relative position recorded in the direct link e.

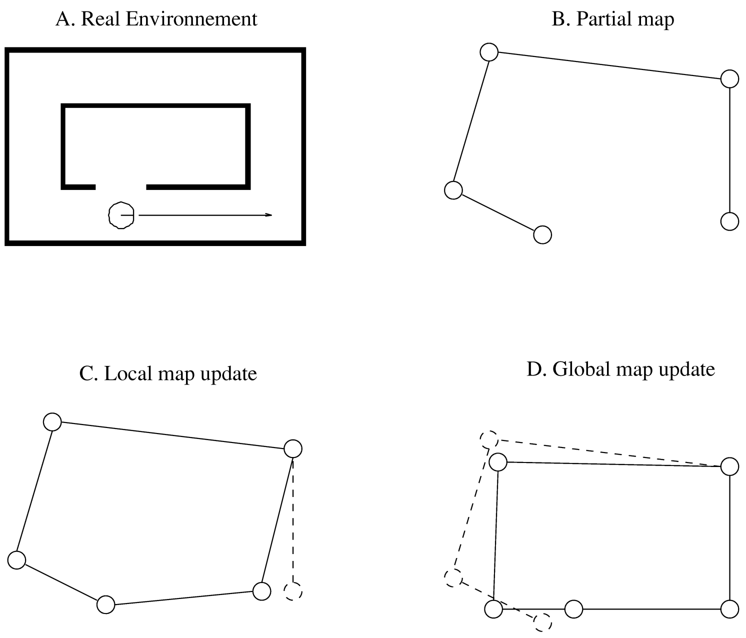

Fig. 12. Illustration of the difference between a local and a global enforcement of consistency. Local update (C) only concerns links that are connected to the current node, while global update potentially modifies all links at each time-step (D).

described in metric map-learning (Section 3.2.3). However, some models take information about past positions into account and globally enforce map consistency and reconsider data association (Fig. 12D), either off-line (Section 3.2.4) or on-line (Section 3.2.5).

Finally, Section 3.2.6 describes the exploration strategies that can be used to constrain the movements of the robot in order to limit map errors.

3.2.1. Adding nodes to the map

When no perceptual aliasing is supposed to occur in the environment, topological mapping is relatively straightforward. As for localization, the robot continuously compares its current allothetic information to data stored in the map's nodes. But, contrary to what is done for localization, the node which is the most similar to the current situation is not always recognized as the current position. Instead, a threshold is used to decide whether the current position corresponds to this node or to a place never-seen-before. If the maximal similarity is above this threshold, the corresponding place is recognized as the robot's current position. Otherwise, this node is not recognized and the robot is considered to be in a new place. Consequently, a new node, characterized by the current perceptions, is added to the map.

When the mapping system has to cope with perceptual ambiguities, this strategy is useful when a never-seen-before sensory situation is encountered, but an already-know sensory situation may correspond either to an already visited place or to a new place. To differentiate between these two cases, idiothetic information has to be taken into account. In models that use position-tracking for localization, the standard localization procedure is used and predicts the robot's current position using idiothetic cues. The procedure used in the case without perceptual aliasing is then applied over the nodes in a limited area around this predicted position, assuming that this area is not subject to perceptual aliasing. When the links of the map simply encode the adjacency of places, this area contains nodes that are connected to the previously recognized node. When metric data are associated with the links, the area can simply correspond to a given distance around the predicted position (Kuipers & Byun, 1991; Engelson & McDermott, 1992; Kurz, 1995; Donnart & Meyer, 1996b; Yamauchi & Beer, 1996; Kunz, Willeke, & Nourbakhsh, 1997; Von Wichert, 1998; Nehmzow & Owen, 2000). Such an area can be adjusted to take into account the robot's position uncertainty, thus making poorer matches possible if the precision of the estimate is low. This can be achieved, for example, using the *Mahalanobis distance*, which takes into account the variances and covariances of the positions (Balakrishnan et al., 1999).

Other systems directly take into account the position information to determine which node on the map best fits the current situation. The corresponding matching depends both on the similarity between perceived and recorded allothetic information and on the closeness between the robot's and the node's positions. Consequently, it is possible to simply add a threshold to the recognition mechanism to be able to decide when new places have to be recorded (Arleo & Gerstner, 2000; Touretzky, Wan, & Redish, 1994; Dedeoglu, Mataric, & Sukhatme, 1999; Mataric, 1992).

As for systems that rely on multiple-hypothesis tracking (Filliat & Meyer, 2003), the decision between recognizing an existing node and creating a new one is more difficult. The main reason is that multiple-hypothesis tracking for localization relies on the comparison of the robot's different possible positions in the map in order to select the one that best corresponds to the available data. Hence the localization algorithm itself cannot work if the robot is outside the mapped area. For this reason, most models that rely on multiple-hypothesis tracking cannot build a map on-line but resort to off-line map-learning (Section 3.2.4). Some models (Hafner, 2000; Filliat & Meyer, 2002), however, are able to track multiple hypotheses and build a map on-line. These models rely on an heuristic that detects whether the robot is currently in the mapped area or not. If it is the case, the standard multiple-hypothesis tracking algorithm is used to determine the robot's position. If not, a new node is added to the map. This heuristic could rely on a simple threshold tied to the most probable position (Hafner, 2000), or could be based on the variation of the sum of the probabilities of all the concurrent position hypotheses (Fig. 13) (Filliat & Meyer, 2002). The latter method allows greater robustness as it explicitly detects when the robot moves out of the mapped area.

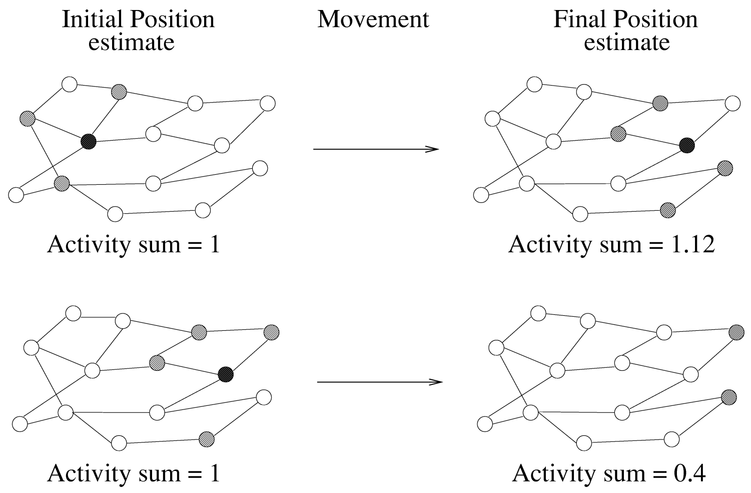

Fig. 13. Illustration of the heuristic used in Filliat and Meyer (2002a) to detect when a node should be added in the map. When the robot moves inside the mapped area (top of the figure), the total sum of the probabilities of the various hypotheses remains approximately constant. In contrast, if the robot exits the mapped area (bottom of the figure), this sum suddenly decreases, making it possible to detect that it is necessary to add a new node to the map.

3.2.2. Odometry correction

In environments where robots face perceptual aliasing issues, idiothetic cues play a major role in localization and map-learning. Consequently, various techniques are used to prevent idiothetic errors growing too high. As these techniques are detailed in a companion review of localization methods (Filliat & Meyer, 2003), they are briefly summarized in this section.

When idiothetic cues are used locally only, i.e. between two connected nodes, and are reset every time the robot detects a new place, odometry precision is not required to be very high. As a consequence, raw odometry can be sufficient for map-learning and some systems do not correct it between places (Kuipers & Byun, 1991; Mataric, 1992; Engelson, 1995; Donnart & Meyer, 1996b). Other systems use an external sensor, such as a compass (Nehmzow & Owen, 2000; Filliat & Meyer, 2003), or assumptions about the environment, such as orthogonal corridors (Dedeoglu et al., 1999; Kunz et al., 1997), to correct the raw odometry. This correction mainly concerns the robot's direction which is usually not estimated, or corrected, for topological localization, but which is often essential to navigation.

When idiothetic cues are used globally, i.e. when a position is associated to each node in a common reference frame, odometry correction is more important, as long-term consistency has to be achieved. But in this case, the positions of the nodes that are already in the map can be used to correct odometry measurements. This situation resembles those of metric map-learning where already-mapped feature perceptions can be used to correct the robot's estimated position. In the topological context, corrections may be made occasionally when the odometry precision is considered insufficient. This entails using special-purpose procedures that estimate the robot's position using the known map of the environment and allothetic cues only (Yamauchi & Beer, 1996; Touretzky et al., 1994; Arleo & Gerstner, 1999). This estimated position is then used as the new reference for subsequent idiothetic cues. Odometry corrections may also be continuously performed by using the recognized node position at each time step. This position may either be taken directly as the new reference position (Mataric, 1992), or it may be used to correct the robot's current position estimate through Kalman filtering (Kurz, 1995; Balakrishnan et al., 1999).

3.2.3. Locally updating link information

When a node has been recognized, or when a new node has been created, idiothetic information gathered since the last node recognition is used to update or to create a connection between these two nodes. This section presents models in which modifications only concern the current connection, and do not propagate to the whole map.

In (Sharp, 1991; Burgess, Donnett, Jeffery, & O'Keefe, 1997; Levitt & Lawton, 1990), nodes are defined by the landmarks that are visible from the corresponding positions. Hence the link between two nodes is implicitly recorded in the definitions, as it is possible to infer the relative positions of two nodes when they share common landmarks.

In (Bachelder & Waxman, 1995), the relation between the previous and the current node is also implicit, because it is encoded in a neural network that predicts which node may be reached from the current node, thus affording the same information as links between nodes.

However, most models explicitly create links between the previous and the current node to indicate that it is possible to go directly from one node to the other. In some models, this connection simply expresses the adjacency (Franz, Scholkopf, Georg, Mallot, & Bulthoff, 1998; Gaussier et al., 1998; Kortenkamp & Weymouth, 1994).

When idiothetic information is memorized in the links, it must be updated after each robot's move. When this information is only used locally, i.e. when the relative positions of distant nodes are not calculated, such an operation is straightforward and may simply call upon a weighted mean of the old and new values (Kuipers & Byun, 1991; Engelson & McDermott, 1992; Kunz et al., 1997; Von Wichert, 1998; Dedeoglu et al., 1999).

However, when the idiothetic information is used globally, i.e. when it is used to compute the relative positions of all the nodes of the map, the consistency of the map has to be maintained. Maintaining this consistency is particularly important when large cycles are traveled through in the environment (Section 3.1.1) and raises the same issue as reconsidering past updates in metric maps (Section 3.1.4), because previous idiothetic data recorded in the map must be coherent with new ones. Global solutions to this problem will be presented in the next two sections, but it is possible to enforce consistency using only local map modification.

A solution to automatically achieve global map consistency without updating every link is to associate a position to each node in a metric framework, instead of associating lengths to the links. Using this technique, the relative position of two nodes is independent of the path leading from one node to the other, and the map is therefore automatically coherent. In these models, map modification are only local and only concern the position of the recognized node. Some models assign a metric position at the node creation and do not change it afterwards (Touretzky et al., 1994; Yamauchi & Langley, 1997; Oore, Hinton, & Dudek, 1997; Arleo & Gerstner, 2000), while other models update it with the robot's current position each time the node is recognized, for example by using a Kalman filter (Balakrishnan et al., 1999) or a simple average of the estimated positions (Mataric, 1992; Donnart & Meyer, 1996b; Kurz, 1995; Duckett & Nehmzow, 1997; Nehmzow & Owen, 2000).

3.2.4. Globally updating links off-line

Map consistency can also be maintained by algorithms that modify all the links of the map and reconsider past data association each time new idiothetic cues are gathered. These methods come down to reconsidering past map updates in the light of new information on the robot's past positions, as mentioned in the metric map-learning section.

As for metric maps, it is possible to use off-line algorithms to generate maps from data sets recorded by the robot so as to be able to use future information about its positions. The corresponding models are based on POMDP models of the environment (Filliat & Meyer, 2003) and use expectation-maximization algorithms in order to build a map (Section 3.1.4). Here again, using an approximate map of the environment together with probabilistic sensor and actuator models, the algorithm produces an enhanced map that maximizes the probability of the training data. However, as mentioned in Section 3.1.4, such algorithms converge to local maxima only. In a topological map-learning context, they are also poorly effective for learning the map structure, i.e. the arrangement of the nodes within the environment. Instead, they are much more effective at learning the parameters of the POMDP states, i.e. the allothetic definition of the nodes. As a consequence, an initial map with an almost correct structure is generally used to enhance the algorithm's efficiency.

The model of Simmons and Koenig (1996) calls upon a topological map of the environment initially provided by an operator. However, this initial map is imprecise and contains uncertainties in corridor lengths. It is converted to a POMDP model, in which nodes represent regularly spaced positions within the environment. As the distances between nodes are fixed, uncertain corridor lengths are modeled by several parallel chains of nodes with different node numbers (Fig. 14). An EM algorithm is then used to learn both the probabilities of making perceptions at each node and the transition probabilities between nodes. Corridor lengths are learned by selecting the node chain that best corresponds to actual idiothetic data. Using some additional hypotheses, like constancy of the transition probabilities between adjacent nodes over the whole environment, the authors are able to produce an improved map using a smaller amount of training data for the EM algorithm.

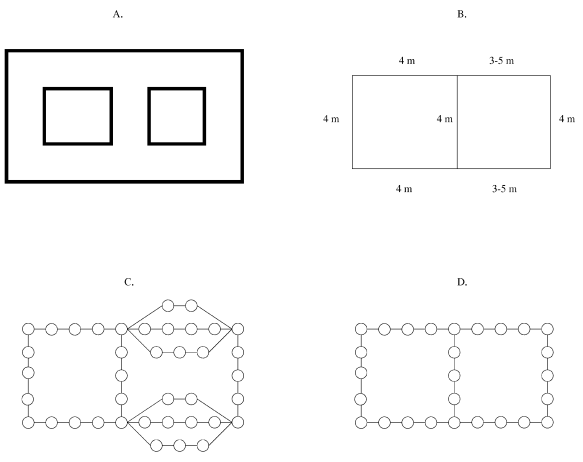

Fig. 14. (A) The real environment the robot is mapping. In the model of Simmons & Koenig (1996), a simple topological representation of the environment, along with some metric information, is initially provided to the mapping system (B). This map is then transformed into a POMDP model of the environment, where states represent positions with 1-m intervals. Several parallel chains are used to model uncertain corridor lengths (C). Expectation-maximization is then used to learn perception probabilities for each node and transition probabilities for each link between nodes. Eventually, chains representing true corridor lengths are selected (D).

Theocharous, Rohanimanesh, & Mahadevan (2001) also start with an a priori given topological map of the environment that is converted to a hierarchical POMDP model. In this model, several levels of states are used that represent the environment with different spatial resolutions. Low-level states, similar to those of Koenig's model, are used to accurately represent the robot's possible positions. Higher-level states make it possible to group low-level states to represent environmental features such as corridors or corners (Fig. 15). The algorithm used to build such a map is EM adapted to hierarchical POMDP, which learns perception probabilities for each low-level state, and transition probabilities between low-level and high-level states. The learning of such hierarchical models achieves performances similar to those of the standard flat models. However, taking advantage of the hierarchical structure of the model makes it possible to separately learn groups of low-level states that represent corridors or junctions. Re-using such partial maps of the environment, and learning to combine them in a global map, affords better and faster learning for subsequent maps.

The approach of Shatkay & Kaelbling (2002) also

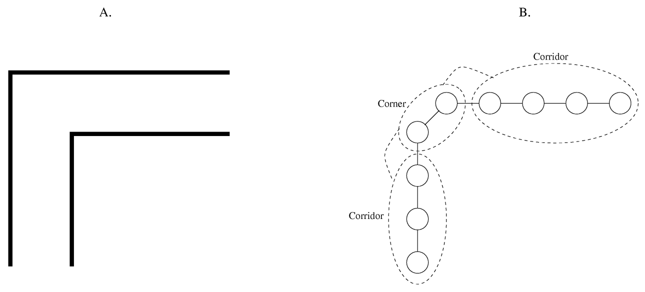

Fig. 15. (A) Actual environment a robot is mapping. (B) Corresponding map in the model of Theocharous et al. (2001), where low-level states represent regularly-spaced possible positions for the robot. Each of these states records the probabilities of the robot's perceptions in the corresponding positions. Links between states correspond to idiothetic cues relating two such positions. These states are grouped in higher-level states that represent structures of the environment such as corridors and corners.

learns topological models using an EM algorithm. However, unlike in the previous models, link lengths are variable and not fixed to a given value. As a consequence, nodes are used to represent junctions and doors along corridors, while links represent corridors. Moreover, the initial model is not provided by a human operator, but is generated from idiothetic data by a clustering of similar inter-states odometry readings. This clustering detects which odometry readings correspond to the same node transition. These transitions are then put together by a *state-tagging* algorithm that associates each transition to a start and end node, thereby creating an initial map. The EM algorithm then learns perception probabilities at each node, together with the lengths and directions of the links. Associating this metric information to the links, instead of using purely topological information, makes it possible to effectively learn topological models of the environment within a POMDP framework.

Using formal logic theory, Remolina & Kuipers (2001) describe a method for building a topological map from a set of idiothetic and allothetic cues. The considered cues consist of a succession of distinctive states (Filliat & Meyer, 2003) and actions that link these states. This set makes a description of the environment at the so-called *causal level*, i.e. it describes the environment by the effects of actions in the different states related to perceptions. However, it does not constitute a map because, owing to perceptual aliasing, a given state may correspond to different places in the environment. Capitalizing on this representation, the model makes it possible to create a topological map by abduction, i.e. by finding the map with the minimal number of places that can explain the observed cues. The logical theory that is used to create this map is based on various axioms that enforce general properties of the map and of spatial behaviors. For example, it exploits the fact that turning leaves the robot at the same place.

3.2.5. Globally updating links on-line

Finally, some models (Duckett, Marsland, & Shapiro, 2000; Hafner, 2000; Filliat & Meyer, 2002) are able to take care of map consistency by globally updating links on-line each time new idiothetic data are gathered. However, these models are not able to reconsider data association and only allow the idiothetic cues stored in the map to be reconsidered globally. The corresponding procedure considers that links of the map are sort of springs, the relaxation length of each corresponding to a link length measured by odometry. These springs being connected in a network according to the map layout, an incoherent map corresponds to a spring network that is not in equilibrium. Therefore, a relaxation algorithm is used to compute the equilibrium state of the network, thus providing a consistent map whose link lengths are closest to the measured lengths. Such a relaxation algorithm only provides a heuristic method for reconsidering past link updates in the presence of new information, but it proves to be efficient and to allow correct map-learning.

3.2.6. Exploration strategies

Usually, the major objective of a given exploration strategy is to guide the robot in order to explore the environment as quickly and as completely as possible. However, within the framework of topological map-learning, as the recognition of a given sensory situation may be subject to errors, some models implement dedicated exploration strategies that aim at checking if the recognition is correct. The corresponding procedures often play an important role in these models because they prevent an accumulation of errors in the map that would lead to instability.

An example of a procedure for recognition verification is the *rehearsal strategy* of Kuipers & Byun (1991) that exploits the expected consequences of some robot moves as suggested in the Introduction. Whenever an already existing node is recognized, the system uses the map to guide the robot towards another already known node which is connected to the recognized node. If this second node is also recognized, this means that the previous recognition was correct and the robot resumes its map-building behavior. Otherwise, it may be concluded that the previous node was erroneously recognized, and a new node is added to the map. To warrant termination, this procedure is not used recursively. As a consequence, two successive recognition errors cannot be detected by the system. However, other strategies are guaranteed to converge if some hypotheses are made about the environment (Dudek, Jenkin, Milios, & Wilkes, 1997; Basye, Dean, & Vitter, 1997). During map-learning, in order to warrant an exhaustive exploration of the environment, an *exploration agenda*, which memorizes detected but untraveled corridors, is also used to direct the robot's movements.

Several other exploration strategies are used by Engelson (1995) and implemented as *opportunity scripts*. Such scripts are short sequences of actions, designed to detect and correct map errors. For example, a *Head for rare waypoints* script computes the robot actions that drive it toward a neighboring node which has not often been encountered, in order to check if this node is not a *transient*, i.e. a node that has been created because of perception errors and that no longer fits the actual environment. Other scripts also direct exploration toward unexplored or uncertain areas, to make the mapping process as exhaustive as possible. Each script has an application condition related to the current states of the map and the robot, and this condition is continuously checked by an *opportunity checker*. Scripts that can be applied in the current situation may then be chosen and executed by a planning process.

Kunz et al. (1997) describe a model that strongly relies on exploration strategies. Their system assumes that corridors are orthogonal in order to predict the layout of the environment when new intersections are detected. These predictions are added to the map as hypotheses and the robot is then guided in order to check these hypotheses. If predicted hallways or intersections are not detected, they are removed from the map; otherwise, they are confirmed.

4. Path-planning

Once a robot is provided with a map, once an estimate of its position inside this map is available, and once a goal position is singularized within this map, the robot should be able to navigate from its current position to the goal position. This path-planning capacity relies on the computation or on the learning of a plan that allows the goal to be reached. The corresponding research domain is wide and relatively far removed from those of localization and map-learning, and it faces very intricate issues when robots with many degrees of freedom are involved. However, relatively simple techniques can be called on for 2D path-planning in autonomous robotics. This article accordingly surveys these techniques without going into detail about the difficulties of path-planning in general (see Latombe, 1991 or Laumond, 1998) for more general reviews). Section 4.1 mentions the specific issues arising during the actual execution of such a plan. Section 4.2 defines the two sorts of plans that are commonly used. Methods for computing or learning these plans are then presented, first in a discretized search-space (Section 4.3), then in a continuous space (Section 4.4).

4.1. Executing a plan

The execution of a plan prescribing movements toward the goal raises a number of issues. The first is concerned with the fact that discrepancies almost always exist between the map and the actual environment. These discrepancies, for example, may be caused by dynamic obstacles such as humans. Consequently, a precise plan generated using an incorrect map may be impossible to execute. A second issue stems from the fact that a robot's actions may be noisy, for instance because of wheel slippage. Here, even when the robot precisely executes a correct plan, the goal may still be missed.

Such issues caused the first robots that used so-called *hierarchical controllers* (Murphy, 2000) to behave poorly in realistic environments. The reason is that these controllers did not have any control over the execution of the plan that was devised (Fig. 16A). A radically opposite approach was brought forth by Brooks (1991), who advocated the suppression whenever possible of internal world models such as environmental maps, and plead for the use of the *behavior-based* approach to robotics. This approach placed emphasis on reactive behaviors that acted in a closed-loop with the environment (Fig. 16B) and that avoided many problems arising with world models, thus leading to efficient, albeit simple, robots.

However, most current robotic systems mix these two opposite approaches by using an *hybrid deliberative/reactive* control architecture (Arkin, 1989, 1998; Murphy, 2000). Within these architectures, a low-level reactive controller is responsible for the execution of moves prescribed by the high-level

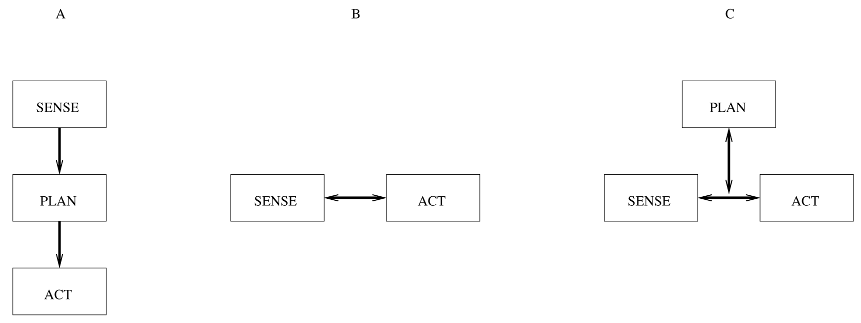

Fig. 16. Various architectures have been devised to control autonomous robots. Earlier architectures were organized in a *hierarchical* fashion. These architectures relied on the elaboration of a correct world-model in which a plan was carried out. The plan was then executed on the real robot without closed-loop control, thus leading to frequent failures (A). The *behavior-based* solution to these problems was to suppress world models and planning by calling upon reactive behaviors alone, which may be more efficient when confronted with unforeseen situations. The corresponding architectures called upon a variety of sensory-motor modules that operated in parallel (B). Most modern robots use a *hybrid* approach, with a high-level controller that models the environment and plans actions, while low-level reactive controllers are responsible for their execution and for reacting to unforeseen situations (C).

controller, while continuously reacting to unforeseen situations such as unmodeled obstacles (Fig. 16C). The high-level controller is responsible for map-learning, localization and path-planning. The interplay of these two levels affords robots the possibility of quickly reacting to their environment, while remaining able to efficiently execute long-term plans.

4.2. Two kinds of plan

In its general sense, a plan is a sequence of actions to be taken to achieve a given goal. In a path-planning context, it consists of a list of moves to be executed sequentially, each such move possibly followed by the detection of a position already stored in the map when the plan is executed in a closed-loop within the environment. According to this view, a plan is a path that is elaborated thanks to a mental exploration of the map, in agreement with the description of Craik (1943) of planning as a series of ‘‘experiments in the head’’, and this exploration is both deterministic and systematic (Fig. 17). Usually, the execution of this kind of plan leads to deterministically following the corresponding path.

However, other authors define planning as a ‘‘goal-directed selection of reactions to possible situations’’ (Schoppers, 1987). According to this view, either a mental exploration of a map or actual moves in the environment, which may both be performed deterministically or at random, lead to the elaboration of what is called a *universal plan*, i.e. an association of the action to be performed to reach the goal with each possible position (Fig. 17). Then, the execution of this kind of plan entails deterministically or probabilistically executing the move associated with each position.

4.3. Path-planning in a discretized space

As previously mentioned, one of the advantages of topological maps is that the representation of the environment as a graph of places makes path-planning relatively straightforward because any classical method for graph-search (Barr & Feigenbaum, 1981) may be used. When metric maps are used, a majority of planning methods start by discretizing the free space available to the robot and then apply similar graph-search techniques.

4.3.1. Discretizing the search space

A first class of methods for discretizing the search space extracts a topological map from a metric map by decomposing the free space into small cells corresponding to topological nodes. Such decomposition may be integrated in the map-learning process, for example by using a map representation that intrinsically decomposes the free space into cells representing the environment (Arleo et al., 1999) (Section 3.1.1). Other models (Chatila & Laumond, 1985; Thrun, 1999; Buhmann et al., 1995) (Section

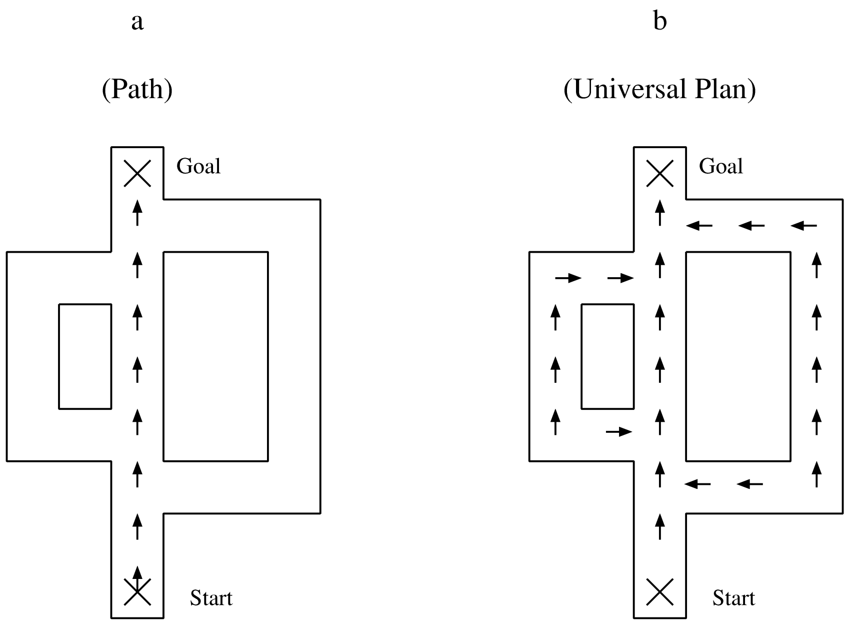

Fig. 17. Two plans for solving the same navigation problem: (a) a single path leading from the start place to the goal place and (b) a universal plan defining several paths to the goal.

3.1.6) integrate such decomposition in the map-learning process by explicitly extracting a topological map from the metric map. But some models use dedicated methods to decompose the free-space for path-planning. These methods may be classified as exact methods, which cover the free space exactly, or as approximate methods, which only represent a part of the free space that is convenient for path-planning, without trying to cover the whole free space. An example of an exact method is the decomposition into convex polygons (Latombe, 1991), also referred to as a *meadow map* (Murphy, 2000). Approximate cell decompositions may call upon rectangles similar to those of Arleo et al. (1999), or upon regular grids or quad-trees (Murphy, 2000) (Fig. 18).

A second class of methods, known as *roadmap methods* (Latombe, 1991), may be used to discretize the search space for path-planning. Instead of decomposing the free space into discrete areas, it is decomposed into a set of partial possible paths, called a *roadmap*, that join a number of *key-points* scattered within the environment. Path-planning therefore entails combining these partial paths so as to create an overall path from the start to the goal positions. The roadmap may derive, for example, from a *visibility graph* (Fig. 19), which includes the straight lines joining obstacle angles that are visible from each other. In this case, the key-points of the roadmaps are the obstacle corners, which make it possible to compute short paths that lie close to the obstacle. However, this closeness to the obstacles can be undesirable, for example if the robot has to move fast. Another method for computing roadmaps, which maximizes clearance from obstacles, calls upon *Voronoï diagrams*. In this case, the key-points of the roadmaps are points equidistant from at least three obstacles (Fig. 19). Still other methods for computing roadmaps may be found in the literature (Latombe, 1991).

Once a suitable decomposition has been achieved, planning a path toward the goal first entails calculating a path between the initial robot position and the closest point in the discretized space. This point must be either a key-point of a roadmap, the center of the cell that contains the initial position, or the middle of a border belonging to this cell. A path is then calculated in the discretized space passing through other points and leading to a point that is close to the goal, according to a method chosen from those described in the next section. When this point has

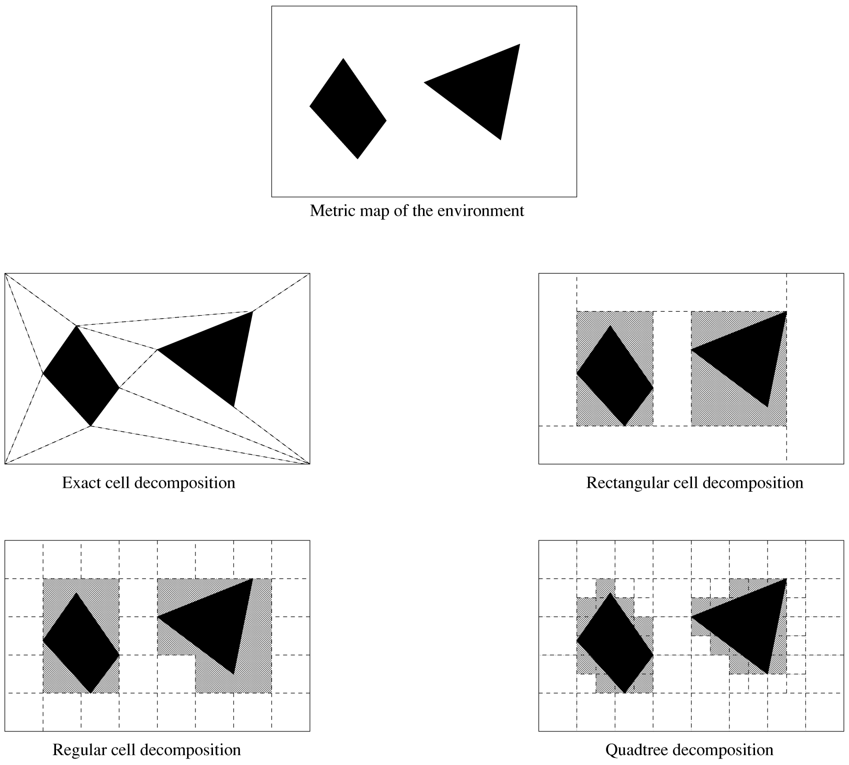
Metric map of the environment
Exact cell decomposition
Rectangular cell decomposition
Regular cell decomposition
Quadtree decomposition

Fig. 18. Examples of cell decompositions used to discretize the path-planning search space. These results call upon either exact or approximate decompositions. Exact decompositions precisely cover the whole free space, while approximate decompositions only cover a part of this space that makes robot movements possible. The latter decompositions rely on various cell types, either rectangles of various sizes, regular squares, or squares of variable size in the case of quad-trees.

been determined, a final move is calculated that leads from this point to the goal. Once the total path has thus been elaborated, it is additionally possible to optimize it, for example using a relaxation method (Murphy, 2000), in order to avoid detours that are side-effects of the chosen discretization (Fig. 20).

4.3.2. Computing a path

Several strategies exist that allow a path to be computed. In particular, classical algorithms for graph-based search, such as Dijkstra's algorithm or the well-known $A^\star$ or its variants (Barr & Feigenbaum, 1981), may be used to calculate the path from the current node to the goal node as is done in (Levitt & Lawton, 1990; Kortenkamp et al., 1994; Kuipers, 2000; Scholkopf & Mallot, 1995; Nourbakhsh, Powers, & Birchfield, 1995; Arleo et al., 1999; Dudek & Jenkin, 2000) for example. In practice, the reasonable size of the maps usually used in robotics affords an efficient use of such algorithms. It is moreover possible to resort to *dynamic programming* (Bellman, 1957) if the map is too large for such a direct approach to be effective. This approach exploits *Bellman's principle*, i.e. the fact that the optimal path from point A to point C via point B is the concatenation of the optimal path from point A to point B, and of the optimal path from point B to point C. Exploiting this property, dynamic programming allows heuristic functions to be used that avoid seeking every possible path leading to the goal from the start place in the map, thus reducing the computational complexity of path-planning.

Computing a path may also be performed by a simple breadth-first search in a graph, starting from the goal. This method is also called *spreading activation* (Mataric, 1992; Bachelder & Waxman,

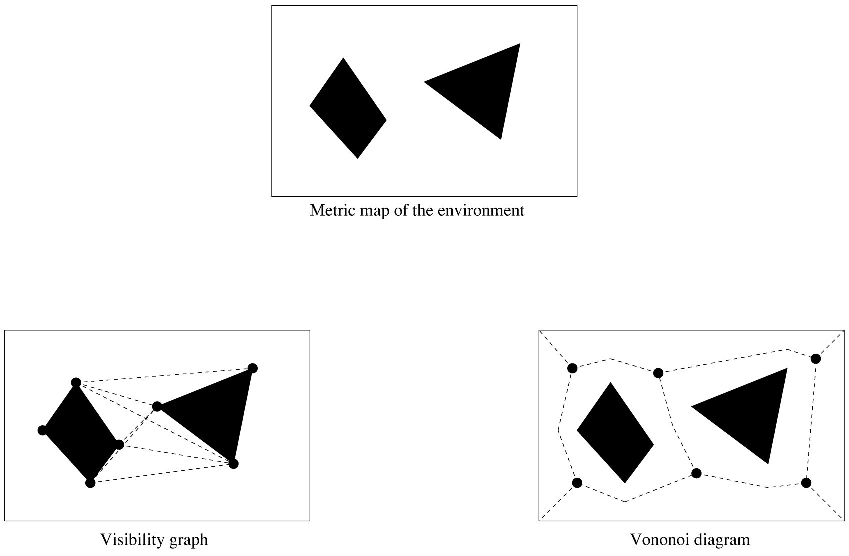

Fig. 19. Two examples of methods used to create road-maps. The first method calls upon a *visibility graph*. In this case, the obstacle’s angles are the key-points, and the straight lines that link two points that are visible from each other are the partial paths. The second method calls upon a *Voronoï diagram*, which is built from the points that are equidistant from several obstacles. In this case, the key-points are the points that are equidistant from at least three obstacles, while partial paths are broken lines that pass through points that are equidistant from two obstacles only and that join these key points.

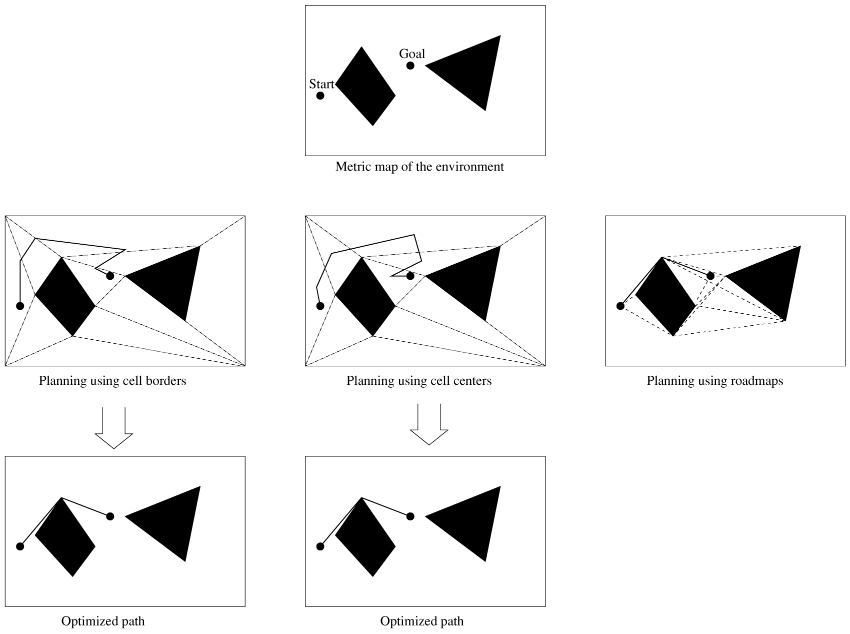
Goal
Start
Metric map of the environment
Planning using cell borders
Planning using cell centers
Planning using roadmaps
Optimized path
Optimized path

Fig. 20. Example of planning in a discretized space for metric maps (see text for details).

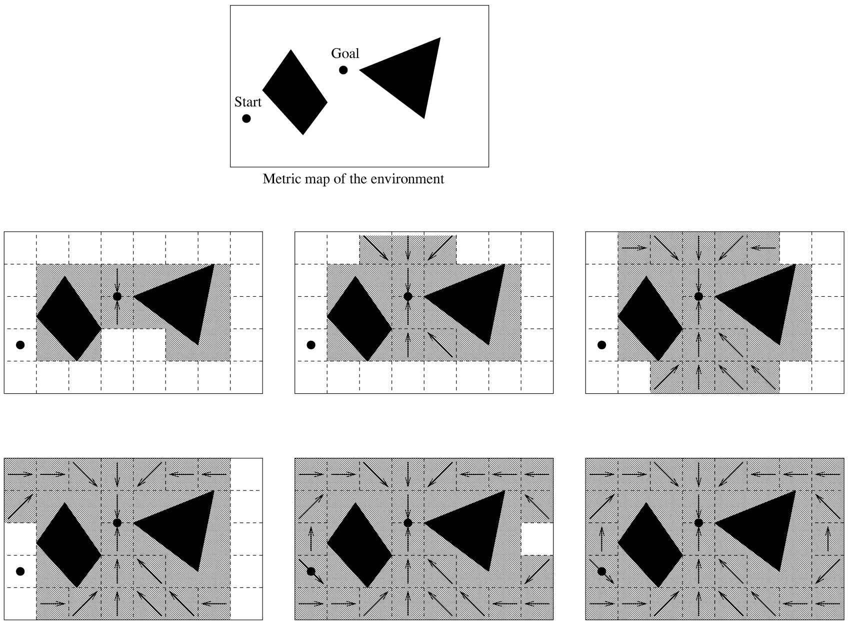

Fig. 21. Illustration of spreading activation planning. Starting from the goal node, the action that guides the robot to the goal is computed for neighboring nodes. This process is then repeated for the neighbors of these latter nodes. Progressively, an action aiming toward the goal is associated with each node of the map.

1995), or *wavefront propagation* (Murphy, 2000). Such expressions are inspired by the fact that the action associated with each node is calculated in an order resembling the progression of a fluid pouring out of the goal (Fig. 21).

A very different approach is described in (Donnart & Meyer, 1996a). In this article, a classifier system is used to describe the map and to control a robot's moves. A set of planning rules decides the robot's current goal, while a set of reactive rules determines its actual moves, and a set of mapping rules manages the on-line building of a topological map with associated metric positions (Filliat & Meyer, 2003). Initially, the robot learns to reach the goal in a straight line. However, if it encounters an unexpected obstacle on its direct route toward the goal, it skirts around it thanks to its reactive rules. Then, the resulting trajectory is analyzed, and so-called *salient states* are characterized along this trajectory that serve to elaborate new planning rules. If the robot needs to reach the same goal again, these planning rules will temporarily change the robot's goal so that, instead of directly aiming at it, with the risk of encountering the previous obstacle again, the robot will now try to pass through the salient states in order to avoid this obstacle from a distance. When the detour is completed and the obstacle is avoided, the robot resumes moving directly toward its overall goal (Fig. 22).

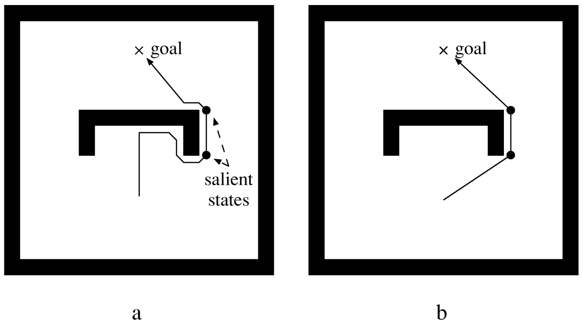

Fig. 22. The planning strategy presented in (Donnart & Meyer, 1996a) calls upon a goal-directed behavior and local obstacle-skirting strategies. After reaching the goal, *salient states* are extracted from the performed trajectory and are used to define new planning rules (a). Subsequently, these new rules will define the salient states as intermediate goals, thus permitting the robot, when starting from the same place and aiming at the same goal, to reach this goal while avoiding the obstacle from a distance (b).

4.3.3. Computing a universal plan

Computing a universal plan, like computing a path, may also be accomplished using breadth-first search in the graph of the map. However, other algorithms may also be used. In the systems described by Buhmann et al. (1995), Thrun (1999) and Burgard et al. (1998), a value V_i is estimated for each node of the map. Initially, the value of the goal node is set to 0, while the value of every other node is set to ∞. The algorithm then iteratively updates each node's value by the smallest value among its neighbors plus a cost of moving between the two nodes. After convergence, the value of a node represents the minimal cost of a path leading from this node to the goal (Fig. 23). Then, the action to perform in each node is the one that leads to the neighboring node with the lowest value.

Quoy, Gaussier, Lepretre, & Revel (1999) present a variant of this method in which the initial value associated with the goal is 1, while the value associated with any other node is 0. This value is updated by the highest value among neighboring nodes multiplied by a coefficient related to the cost of actually moving from one node to the other. Because this coefficient may be different for each pair of nodes, it is thus possible to encode that it is more or less easy to travel between two specific nodes according to the local properties of the environment. The action associated with each node is then the action that goes to the neighboring node with the highest value. As a result, the corresponding trajectory will avoid areas where node transitions are more difficult.

In the model of Burgess et al. (1997), which is inspired from neurosciences and from the way spatial information is encoded in a rat's brain, a robot's position is monitored through *population vector coding* (Georgopoulos, Schwartz, & Kettner, 1986), i.e. through the computation of a mean position stemming from the activities of numerous *place cells*. Each such cell somehow expresses the chances of the robot's being in a given place in the environment. When the robot reaches a goal place, a one-shot learning procedure is triggered that associates with each place-cell active in front of the robot the reverse direction of the robot, i.e. the direction in

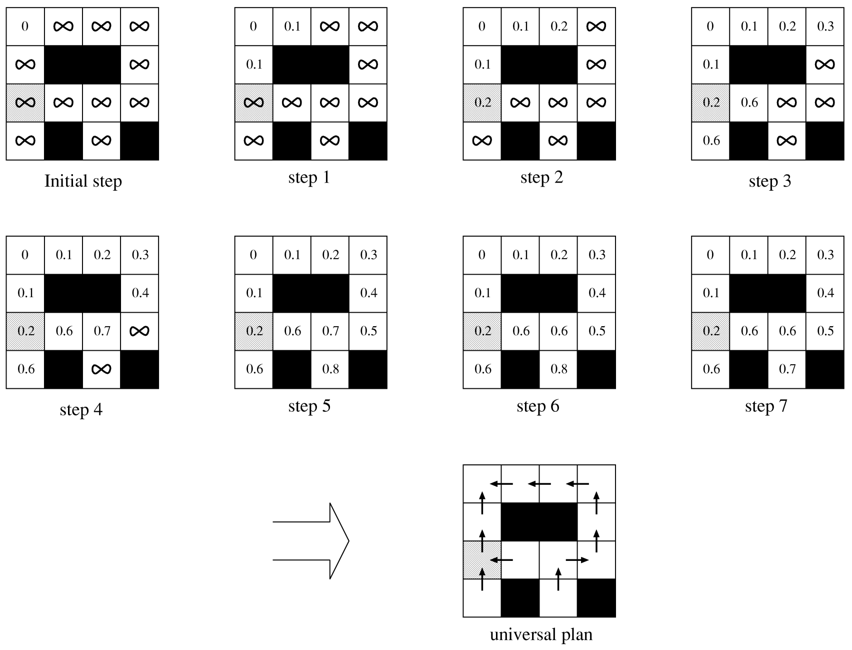

Fig. 23. Illustration of the algorithm of Thrun (1999) for computing a universal plan. In this example, moving to a black cell is impossible, moving to a white cell is associated with a cost of 0.1, while moving to a grey cell is associated with a cost of 0.4. After convergence of the algorithm, the action associated with each cell is the one leading to the neighboring cell with the smallest value.

which to move to get to the goal from the considered cell. However, it should be noted that this approach does not work in environments cluttered with obstacles, thus making it applicable in open areas only.

Models described so far assume that the robot's position is known precisely. However, as by Filliat & Meyer (2003), some models, mostly based on POMDPs, estimate a probability distribution over the possible positions. Using these probabilities to choose the action to be executed affords additional robustness to the planning process.

A priori, an optimal plan may be derived from an underlying POMDP model. This would entail associating an action, not with each node of the map, but with each probability distribution over map nodes. As a result, a robot could temporarily choose moving away from its goal, should these actions reduce uncertainty in the position estimates and globally result in a more efficient goal-reaching behavior. However, implementing this strategy is computationally extremely difficult and cannot be achieved in practice if the map has more than a few dozen nodes.

The probability distribution may nevertheless be taken into account by different means. A first method is to associate an action with each node in the map as in the deterministic case, and then to select the action to be carried out according to the probability distribution of the robot's positions. For example, Simmons & Koenig (1995) simply use the $A^{\star}$ algorithm to associate an action with each state of the POMDP model of the environment. The action to be performed is then chosen according to a *voting scheme*. Given the current probability distribution representing the robot's positions, a *probability mass* is computed for each possible action as the sum of the probabilities of the states that are associated with this action. The chosen action is then the action with the highest probability mass. Cassandra, Kaelbling, & Kurien (1996) present a similar method, except that the corresponding plan is estimated using a probabilistic procedure instead of a deterministic one.

Filliat & Meyer (2002) also rely on a similar scheme, using a universal plan determined by a simple breadth-first search algorithm. An additional mechanism is implemented to continuously check the progression of the robot in order to detect potential failures to reach the goal. If the progression is stopped before the goal is reached, the nodes of the map surrounding the current position are excluded from the planning process and the universal plan is recalculated. This results in a trajectory that avoids the area where progression is impossible and represents an alternative route to the goal.

Although these approaches only lead to actions that move the robot toward the goal, more sophisticated strategies are implemented in Cassandra et al. (1996), which specify uncertainty-reducing actions. These strategies are based on an estimate of the degree of uncertainty attached to the current representation of the position, which depends on the entropy of the probability distribution representing the position. The lower the value, the more certain the position. Accordingly, the chosen action is an action that moves toward the goal when the entropy is low, or an action that reduces the entropy if it is high. Roy & Thrun (2000) provides another example of a similar strategy, named *coastal navigation*. This expression stems from the fact that the resulting strategy favors paths that pass near obstacles, rather than paths that cross large open areas because, as for a boat, these paths are easier to follow and less prone to localization errors.

4.3.4. Learning a universal plan

Learning a universal plan may be performed in the broader context of reinforcement learning (Sutton & Barto, 1998). Instead of deterministically scanning a map in order to evaluate every possible move, this approach calls upon a series of random explorations, either of the environment or of the map, and incrementally elaborates statistics about the chances that a specific move in a specific place will ultimately lead to the goal, i.e. to a place where a reward is obtained. A particular instance of such a strategy entails computing, for each state-action pair (s,a), a value $Q(s,a)$ that corresponds to the maximal expected reward the robot could gain when taking action a in state s. Once such a function has been learned, an exploitation strategy may be derived, like choosing the action associated with the highest expected reward in each possible robot's state.

In the context of path-planning, in the biomimetic model of Arleo & Gerstner (2000) for instance, the states s correspond to the robot's positions, while the

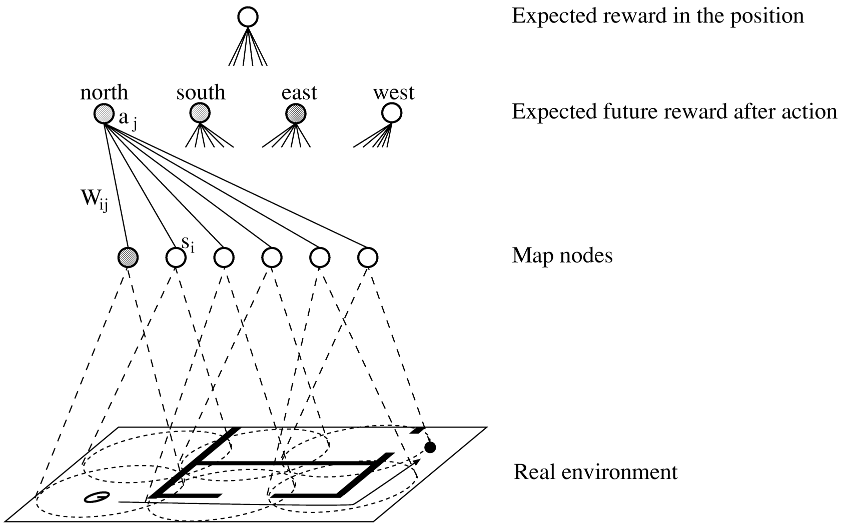
Expected reward in the position
north
south
east
west
a j
Expected future reward after action
Wij
si
Map nodes
Real environment

Fig. 24. Illustration of the structure of the model of Arleo & Gerstner (2000). Four action nodes are connected with the map nodes, each evaluating the reward to be expected when the corresponding action is performed at the corresponding place. The learning process modifies the W_{ij} connections so that such evaluations become increasingly accurate over time. An additional node allows it to learn the reward gained at the current position, so as to be able to detect if the goal subsequently changes.

possible actions *a* correspond to going north, east, south or west (Fig. 24). A reward is associated with the goal cell. After learning, the robot chooses at each step the action that corresponds to the highest expected reward. In this case, learning a plan entails wandering in the actual environment until reaching the goal and receiving a reward. The expected reward $Q(s,a)$ is then updated for all the positions encountered before reaching the goal using the *Q-learning* algorithm (Sutton & Barto, 1998).

Moreover, when the reward is assigned to a different goal, the robot must detect that its goal has changed in order to change its plan accordingly. This is achieved in Arleo and Gerstner's model by a mechanism that learns which reward should be experienced in each position of the map. This mechanism therefore makes it possible to detect when this reward has changed, and when to restart the learning process. It seems to have a neurophysiological counterpart in dopaminergic neurons found in mammalian brains that might detect when an actual reward does not match the expected one (Shultz, Dayan, & Montague, 1997).

4.4. Planning in continuous space

When metric maps are used, it is possible to plan directly in continuous space, without resorting to discretization. Instead, a function that indicates where to move to reach the goal is calculated in each point of the free space. This function provides information similar to universal plans mentioned in previous sections.

The basic tool to implement such a method is the *potential field* (Latombe, 1991; Arkin, 1998). In this approach, the function that is calculated at each point is the sum of an attractive potential emitted by the goal and of repulsive potentials emitted by obstacles. A gradient-ascent strategy is applied to the resulting potential, thus guiding the robot to the goal. However, this basic scheme often produces functions that exhibit local minima, thus preventing the robot from reaching the goal in every situation. Additionally, this scheme may lead to oscillatory movements in the presence of specific obstacles (Koren & Borenstein, 1991b).

Several methods have been devised to avoid these local minima issues. It is for example possible to use the potential field approach to guide a higher-level planner (Latombe, 1991) whose task is to avoid the local minima. Another technique, named *randomized path planner*, triggers a random walk when a local minimum is reached, until the robot escapes from it, before reverting to gradient ascent (Latombe, 1991).

Another possible strategy to avoid these issues is to design alternative potential functions that do not exhibit local minima. This is, for example, the case with *harmonic potential field* functions, i.e. functions that are the solution of Laplace's equation (Dudek & Jenkin, 2000). However, determining such a function is computationally difficult, which makes this approach applicable to small environments only.

5. Discussion

To be able to purposively navigate in an initially unknown environment is a complex task for an animal or a robot, and this task raises numerous issues of perception, categorization and motor control that must all be solved in an integrated manner to ensure survival. Among the different ways an animat may categorize its perceptions to build an internal representation of its environment, and then to use this representation for navigation, some are clearly better suited than others, depending upon the characteristics of the environment, the nature and reliability of the allothetic or idiothetic sensors that the animat may call upon, and the possibilities it has to move itself or to perceptually scan the surroundings.

Likewise, among the different ways an animat may react to circumstances jeopardizing the successful achievement of a map suitable for navigation, some are clearly more appropriate than others, depending upon the circumstances in question, upon the internal representation that is used, and upon the navigation strategy that is implemented. Among such circumstances, the case of errors during the localization process has been dealt with in (Filliat & Meyer, 2003), while that of errors during the incremental building of a map has been discussed earlier in this paper. In both cases, even if general techniques are reused in various systems, each corrective procedure thus implemented was tailored to a specific robot, with its specific sensory-motor equipment, and to the specific kind of map it was building. In other words, because integrated solutions must be sought and implemented, it is highly probable that any such corrective procedure would simply not work with another robot or another type of map. This is clearly the case when strong assumptions are made on the environment (for example the assumption of orthogonal walls), but is also often the case with general algorithms that rely on so many aspects of the whole system that the individual effect of each particular aspect cannot be fully controlled.

Once a reliable map of the environment has been learned, planning procedures may be triggered and a trajectory leading from the current place to its goal may be followed by the animat. However, various additional circumstances may still jeopardize the successful achievement of this latter task. In particular, some unexpected obstacle or specific danger may be encountered that must activate a detour procedure.

In such circumstances, an animat only able to derive from its mapping and planning capacities the place in which it is located and the action it must perform in order to get closer to its goal would resort to a mere place recognition-triggered response navigation strategy (Trullier et al., 1997). Because explicit links between map positions are not exploited by this strategy, the animat cannot anticipate to which new place a specific move from the current place will lead, and therefore it cannot experiment with the consequences of specific moves in its head. Consequently, if a danger or an obstacle is encountered while moving, then the animat is committed to resorting to low-level local-avoidance procedures to skirt around the corresponding zone, thus incurring the risk of reaching unknown places, with which no dedicated goal-seeking actions are attached. The animat will survive or fulfill its mission if its skirting behavior drives it to a place already recorded in its map, from which the right move to resume goal-seeking is already known (Fig. 25). Referring to the Introduction, this strategy only calls upon past and present information to ensure survival. Again, its ultimate success depends upon the

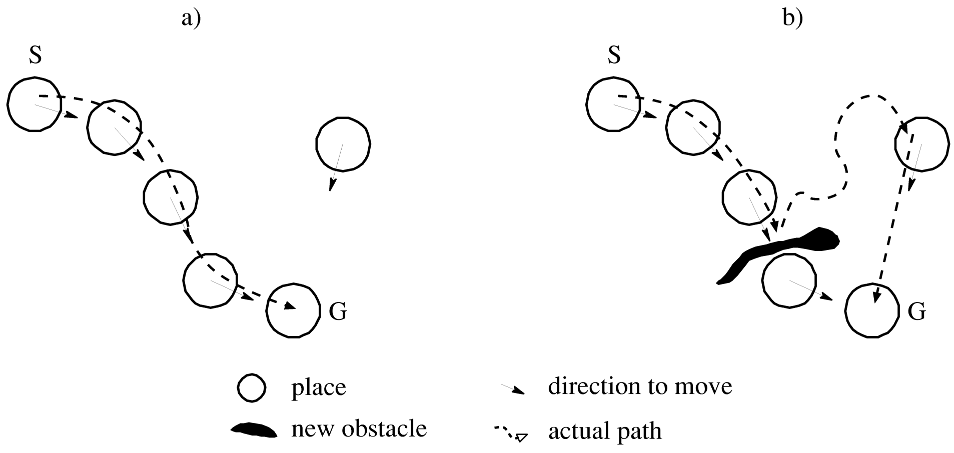

Fig. 25. Navigation according to a Place Recognition-Triggered Response strategy calls upon places and directions recorded in an animat's cognitive map. a) Starting from place S and moving in the associated direction, the animat will successively reach other places from which it will be directed toward the goal place G. b) If an obstacle is encountered on its way, the animat has to wander around until it reaches a known place from which it will be directed toward the goal again (after Trullier et al., 1997).

animat's allothetic sensors and the degree of perceptual aliasing experienced in the environment, the quality of the idiothetic sensors used to monitor the animat's odometry, and more generally the integrated approach that has been used to build its map. Two systems illustrating these points are those of Arleo & Gerstner (2000) and Burgess et al. (1997).

In contrast, if an animat resorting to a topological navigation strategy (Trullier et al., 1997) and following a planned path encounters an unexpected obstacle or danger, it may exploit the links of its topological map to compute from scratch a new path leading to its goal (Fig. 26). This strategy takes into account not only the current situation, but also

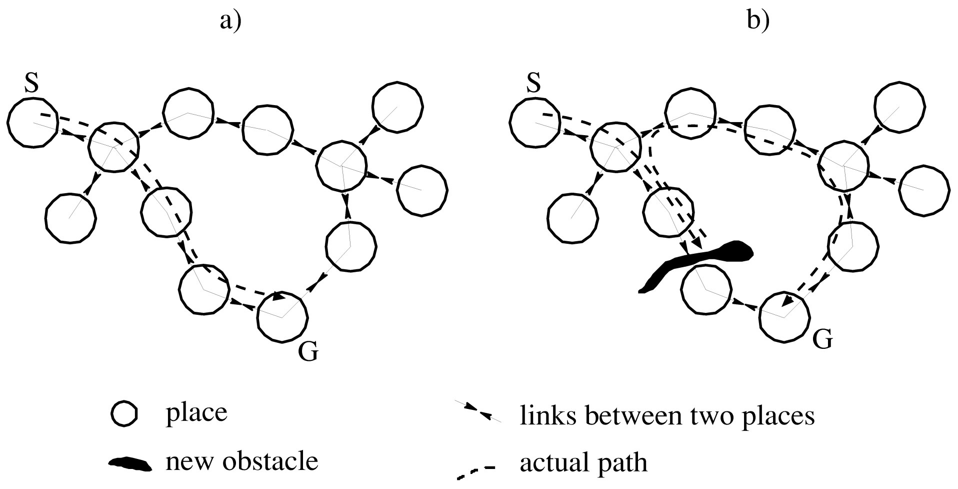

Fig. 26. Navigation according to a topological strategy calls upon places and links recorded in an animat's cognitive map. (a) Starting from place S and following successive links, the animat will reach the goal place G. (b) If an obstacle is encountered on its way, the animat may plan and follow an alternative path to the goal. However, such an alternative path can only pass through already known places (after Trullier et al., 1997).

recorded places and links, on the one hand, and expected consequences of specific moves, on the other hand. In other words, it combines the past, the present and the future at once. However, it should be noted that the detours it generates (Tolman & Honzik, 1930) necessarily pass through places already recorded in the animat's map. Naturally, should an animat following such a detour nevertheless get lost for whatever reason, it could also resort to mere chance to reach its goal as mentioned above. Examples of models resorting to topological navigation are those of Filliat & Meyer (2002), Simmons & Koenig (1995) and Nourbakhsh et al. (1995).

Still more powerful adaptive capacities are afforded to animats relying on a metric navigation strategy (Trullier et al., 1997) that makes it possible to plan detours or shortcuts passing through places that the animat has never encountered before (Fig. 27). Such capacities are afforded by vector manipulations, which are possible within the framework of metric map and space. Here again, experiences in the head and evaluations of future consequences are possible, and the corresponding procedures may also be complemented by mere chance. Metric navigation has been implemented in the systems described by Donnart & Meyer (1996b), Arleo et al. (1999) and Yamauchi et al. (1998) for example.

6. Conclusion

Numerous map-learning and path-planning strategies implemented on autonomous robots have been described in this article. By necessity, each such implementation was carefully tailored to the specific robot used, notably to the capacities and limitations of its sensory-motor equipment, and to the specific environment experienced. Without such careful integration, no navigation system stands any chance of ensuring an animal's survival or a robot's mission under changing and unpredictable circumstances. A hierarchy of navigation strategies has been outlined, together with the sort of adaptive capacities each affords to cope with such circumstances. Animats resorting to a mere place recognition-triggered response strategy essentially capitalize on reflexes and chance to reach their goal when encountering an unexpected obstacle or danger. Animats resorting to a topological navigation strategy may mentally plan a detour and take into account the expected consequences of their acts, but such detours will pass through places already recognized. Lastly, the detours that may be planned by animats resorting to a metric navigation strategy may entail passing through new places, not yet recorded in the animat's internal map.

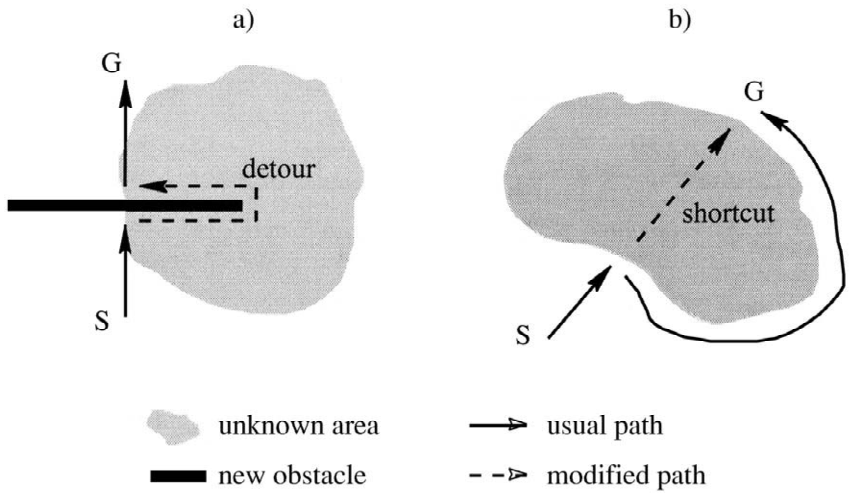

Fig. 27. Navigation according to a metric strategy makes metric detours and metric shortcuts possible. (a) If an obstacle prevents direct access to the goal G, an animat may plan a metric detour to get to it. (b) To shorten its path to the goal, an animat may plan a metric shortcut. In both cases, such detours and shortcuts may pass through yet unknown places (after Trullier et al., 1997).

Acknowledgements

The authors are indebted to anonymous reviewers who greatly helped improving the article. This work was supported by Robea, an interdisciplinary program of the French Centre National de la Recherche Scientifique.

References

Arkin, R. (1989). Towards the unification of navigational planning and reactive control. In *Proceedings of the AAAI spring symposium on robot navigation*. AAAI Press, pp. 1–5.

Arkin, R. (1998). *Behavior-based robotics*. MIT Press.

Arleo, A., Millán, J., & Floreano, D. (1999). Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. *IEEE Transactions on Robotics and Automation*, *15*(6), 990–1000.

Arleo, A., & Gerstner, W. (1999). Neuro-mimetic navigation systems: a computational model of the rat hippocampus. In Drogoul, A., & Meyer, J. A. (Eds.), *Intelligence artificielle située*. Hermès, pp. 193–211.

Arleo, A., & Gerstner, W. (2000). Spatial cognition and neuromimetic navigation: a model of hippocampal place-cell activity. *Biological Cybernetics*, *83*, 287–299, Special Issue on Navigation in Biological and Artificial Systems.

Ayache, N., & Faugeras, O. (1989). Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, *5*(6), 804–819.

Bachelder, I. A., & Waxman, A. M. (1995). A view-based neurocomputational system for relational map-making and navigation in visual environments. *Robotics and Autonomous Systems*, *16*, 267–298.

Balakrishnan, K., Bousquet, O., & Honavar, V. (1999). Spatial learning and localization in rodents: a computation model of the hippocampus and its implications for mobile robots. *Adaptive Behavior*, *7*(2), 173–216.

Barr, A., & Feigenbaum, E. A. (1981). *The handbook of artificial intelligence*. Pitman.

Basye, K., Dean, T., & Vitter, J. S. (1997). Coping with uncertainty in map-learning. *Machine Learning*, *29*(1), 65–88.

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press.

Borghi, G., & Brugali, D. (1995). Autonomous map-learning for a multi-sensor mobile robot using diktiometric representation and negotiation mechanism. In *Proceedings of the international conference on advanced robotics (ICAR-95)*. IEEE Press, pp. 521–528.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, *47*, 139–159.

Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., & Thrun, S. (1995). The mobile robot rhino. *AI Magazine*, *16*(1), 31–38.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., & Thrun, S. (1998). The interactive museum tour-guide robot. In *Proceedings of the fifteenth national conference on artificial intelligence (AAAI-98)*. MIT Press.

Burgard, W., Fox, D., Jans, H., Matenar, C., & Thrun, S. (1999). Sonar-based mapping of large-scale mobile robot environments using EM. In *Proceeding of the international conference on machine learning (ICML-99)*. ProBook.

Burgess, N., Donnett, J., Jeffery, K., & O'Keefe, J. (1997). Robotic and neuronal simulation of the hippocampus and rat navigation. *Philosophical Transactions of the Royal Society B*, *352*, 1535–1543.

Cassandra, A. R., Kaelbling, L. P., & Kurien, J. A. (1996). Acting under uncertainty: discrete Bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems*. IEEE Press, pp. 963–972.

Castellanos, J. A., Montiel, J. M. M., Neira, J., & Tardos, J. D. (1999). The SPmap: a probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, *15*(5), 948–953.

Chatila, R., & Laumond, J. (1985). Position referencing and consistent world modelling for mobile robots. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-85)*. IEEE Press, pp. 138–170.

Cliff, D., Husbands, P., Meyer, J. A., & Wilson, S. W. (Eds.), (1994). *From animals to animats 3, Proceedings of the third international conference on simulation of adaptive behavior*. MIT Press/Bradford Books.

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

Craik, K. (1943). *The nature of explanation*. Cambridge University Press.

Dedeoglu, G., Sukhatme, G. S., & Matarić, M. (1999). Incremental, on line topological map building with a mobile robot. In *Proceedings of mobile robots XIV*, SPJE 99, Boston, September, pp. 129–139.

Donnart, J. Y., & Meyer, J. A. (1996b). Spatial exploration, map learning, and self-positioning with monalysa. In *From animals to animats 4, Proceedings of the fourth international conference on simulation of adaptive behavior (SAB-96)*. MIT Press, pp. 204–213.

Donnart, J. Y., & Meyer, J. A. (1996a). Learning reactive and planning rules in a motivationally autonomous animat. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, *26*(3), 381–395.

Dubois, D., & Prade, H. (1986). *Possibility theory: an approach to computerized processing of uncertainty*. Plenum.

Duckett, T., Marsland, S., & Shapiro, J. (2000). Learning globally consistent maps by relaxation. In *Proceedings of the international conference on robotics and automation (ICRA-2000)*. IEEE Press, pp. 3841–3846.

Duckett, G., & Nehmzow, U. (1997). Experiments in evidence based localisation for a mobile robot. In *Proceedings of the AISB 97 workshop on spatial reasoning in animals and robots*. Springer.

Dudek, G., & Jenkin, M. (2000). *Computational principles of mobile robotics*. Cambridge University Press.

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1997). Map validation and robot self-location in a graph-like world. *Robotics and Autonomous Systems*, *22*(2), 159–178.

Engelson, S. P. (1995). *Continuous map learning for mobile robots*, in The 3rd French-Israeli symposium on robotics, Herzeliah, Israel, May 1995.

Engelson, S. P., & McDermott, D. V. (1992). Error correction in mobile robot map-learning. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-92)*. IEEE Press, pp. 2555–2560.

Feder, H., Leonard, J., & Smith, C. (1999). Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, *18*(7), 650–668.

Filliat, D., & Meyer, J. A. (2002). Global localization and topological map learning for robot navigation. In *From animals to animats 7, The seventh international conference on simulation of adaptive behavior (SAB02)*.

Filliat D., & Meyer, J.A. (2003). Map-based navigation in mobile robots: I. A review of localization strategies. *Cognitive Systems Research*, to appear. Xref: doi: 10.1016/51389-0417(03)00008-1.

Franz, M., Scholkopf, B., Georg, P., Mallot, H., & Bulthoff, H. (1998). Learning view graphs for robot navigation. *Autonomous Robots*, *5*, 111–125.

Gasós, J., & Martín, A. (1997). Mobile robot localization using fuzzy maps. In Martin, T., & Ralescu, A. (Eds.), *Fuzzy logic in AI – selected papers from the IJCAI'95 workshop*. Springer, pp. 207–224, No. 1188 in LNCS.

Gaussier, P., Leprêtre, S., Joulain, C., Revel, A., Quoy, M., & Banquet, J. P. (1998). Animal and robot learning: experiments and models about visual navigation. In *Proceedings of the seventh European workshop on learning robots*. Springer.

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, *233*, 1416–1419.

Gutmann, J., & Konolige, K. (2000). Incremental mapping of large cyclic environments. In *Proceedings of the IEEE international symposium on computational intelligence in robotics and automation (CIRA-2000)*. IEEE Press.

Hafner, V. V. (2000). Learning places in newly explored environments. In *From animals to animats 6*, *Proceedings of the sixth international conference on simulation of adaptive behavior (SAB2000)*. ISAB, pp. 111–120, Proceedings Supplement.

Hallam, B., Floreano, D., Hallam, J., Hayes, G., & Meyer, J. A. (Eds.), (2002). *From animals to animats 7, The seventh international conference on simulation of adaptive behavior (SAB02)*. MIT Press.

Hébert, P., Betgé-Brezetz, S., & Chatila, R. (1995). Probabilistic map learning: necessity and difficulties. In Dorst, L., van Lambalgen, M., & Voorbraak, F. (Eds.), *Reasoning with uncertainty in robotics*. Springer, pp. 307–321, No. 1093 in LNCS.

Hébert, P., Betgé-Brezetz, S., & Chatila, R. (1996). Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: the use of local maps. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-1996)*. IEEE Press, pp. 757–764.

Hughes, K., & Murphy, R. R. (1992). Ultrasonic robot localization using Dempster-Shafer theory. In *SPIE stochastic methods in signal processing, image processing, and computer vision*. Society of Photo Optical, invited session on applications for vision and robotics.

Koren, Y., & Borenstein, J. (1991a). Histogrammic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, *7*(4), 535–539.

Koren, Y., & Borenstein, J. (1991b). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE international conference on robotics and automation*. IEEE Press, pp. 1398–1404.

Kortenkamp, D., Huber, M., Koss, F., Belding, W., Lee, J., Wu, A., Bidlack, C., & Rogers, S. (1994). Mobile robot exploration and navigation of indoor spaces using sonar and vision. In *Proceedings of the AIAA/NASA conference on intelligent robots in field, factory, service, and space (CIRFFSS 94)*. AIAA, pp. 509–519.

Kortenkamp, D., & Weymouth, T. (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the twelfth national conference on artificial intelligence (AAAI-94)*. MIT Press, pp. 979–984.

Kuipers, B. J. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, *119*, 191–233.

Kuipers, B. J., & Byun, Y. T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, *8*, 47–63.

Kunz, C., Willeke, T., & Nourbakhsh, I. (1997). Automatic mapping of dynamic office environments. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-97)*, Vol. 2. IEEE Press, pp. 1681–1687.

Kurz, A. (1995). Alef: an autonomous vehicle which learns basic skills and construct maps for navigation. *Robotics and Autonomous Systems*, *14*, 172–183.

Latombe, J. C. (1991). *Robot motion planning*. Boston: Kluwer.

Laumond, J. -P. (1998). In *Robot motion planning and control*, *Lectures notes in control and information sciences*, Vol. 229. Springer.

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

Leonard, J. J., & Feder, H. (1999). A computationally efficient method for large-scale concurrent mapping and localization. In *Proceedings of the ninth international symposium on robotics research*. Springer.

Levitt, T. S., & Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence*, *44*, 305–360.

Lu, F., & Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, *4*, 333–349.

Maes, P., Mataric, M., Meyer, J. A., Pollack, J., & Wilson, S. W. (Eds.), (1996). *From animals to animats 4*, *Proceedings of the fourth international conference on simulation of adaptive behavior*. MIT Press/Bradford Books.

Mataric, M. J. (1992). Integration of representation into goaldriven behaviour-based robots. *IEEE Transactions on Robotics and Automation*, *8*(3), 304–312.

McFarland, D., & Bösser, T. (1993). *Intelligent behavior in animals and robots*. MIT Press.

McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions*. Wiley.

Mel, B. W. (1995). Animal behavior in four components. In Roitblat, H., & Meyer, J. A. (Eds.), *Comparative approaches to cognitive science*. MIT Press.

Meyer, J. A., Berthoz, A., Floreano, D., Roitblat, H., & Wilson, S. W. (Eds.), (2000). *From animals to animats 6, Proceedings of the sixth international conference on simulation of adaptive behavior*. MIT Press.

Meyer, J. A., Roitblat, H., & Wilson, S. W. (Eds.), (1993). *From animals to animats 2, Proceedings of the second international conference on simulation of adaptive behavior*. MIT Press/Bradford Books.

Meyer, J. A., & Wilson, S. W. (Eds.), (1991). *From animals to animats, Proceedings of the first international conference on simulation of adaptive behavior*. MIT Press/Bradford Books.

Moravec, H., & Elfes, A. (1985). High resolution maps from wide angular sensors. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-85).* IEEE Press, pp. 116–121.

Moutarlier, P., & Chatila, R. (1990). An experimental system for incremental environment modeling by an autonomous mobile robot. In *Experimental robotics 1.* Springer, pp. 327–346.

Murphy, R. R. (2000). *Introduction to AI robotics*. MIT Press.

Nehmzow, U., & Owen, C. (2000). Robot navigation in the real world: experiments with Manchester's fortytwo in unmodified, large environments. *Robotics and Autonomous Systems, 33*(4), 223–242.

Nourbakhsh, I., Powers, R., & Birchfield, S. (1995). Dervish, an office navigating robot. *AI Magazine*, *16*(2), 53–60.

Oore, S., Hinton, G., & Dudek, G. (1997). A mobile robot that learns its place. *Neural Computation*, *9*, 683–699.

Pfeifer, R., Blumberg, B., Meyer, J. A., & Wilson, S. W. (Eds.), (1998). *From animals to animats 5, Proceedings of the fifth international conference on simulation of adaptive behavior*. MIT Press/Bradford Books.

Quoy, M., Gaussier, P., Lepretre, S., & Revel, A. (1999). A neural model for the visual navigation and planning of a mobile robot. In *Proceedings of the fifth European conference on artificial life—ECAL99.* Springer.

Remolina, E., & Kuipers, B. (2001). A logical account of causal and topological maps. In *Proceedings of the seventeenth international joint conference on artificial intelligence (IJCAI-01).* Morgan Kaufman.

Roy, N., & Thrun, S. (2000). Coastal navigation for mobile robots. *Advances in Neural Information Processing Systems, 12*, 1043–1049.

Scholkopf, B., & Mallot, H. A. (1995). View-based cognitive mapping and path planning. *Adaptive Behavior*, *3*(3), 311–348.

Schoppers, M. J. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th international joint conference on artificial intelligence (IJCAI'87).* William Kaufmann, pp. 1039–1046.

Sharp, P. E. (1991). Computer simulation of hippocampal place cells. *Psychobiology*, *19*(2), 103–115.

Shatkay, H., & Kaelbling, L. P. (2002). Learning geometrically-constrained hidden Markov models for robot navigation: bridging the topological–geometrical gap. *Journal of Artificial Intelligence Research*, *16*, 167–207.

Shultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593–1599.

Simmons, R., & Koenig, S. (1995). Probabilistic navigation in partially observable environments. In *Proceedings of IJCAI'95.* Morgan Kaufman, pp. 1080–1087.

Simmons, R., & Koenig, S. (1996). Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-1996).* IEEE Press, pp. 2301–2308.

Smith, R., Self, M., & Cheeseman, P. (1988). Estimating uncertain spatial relationships in robotics. In *Uncertainty in artificial intelligence*. Elsevier, pp. 435–461.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. MIT Press.

Theocharous, G., Rohanimanesh, K., & Mahadevan, S. (2001). Learning hierarchical partially observable Markov decision processes for robot navigation. In *Proceedings of the IEEE conference on robotics and automation (ICRA-2001).* IEEE Press.

Thrun, S. (1999). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*(1), 21–71.

Thrun, S., Burgard, W., & Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-2000).* IEEE Press, pp. 321–328.

Thrun, S., Fox, D., & Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, *31*, 29–53, joint issue with *Autonomous Robots 5.*

Todd, P., Wilson, S., Somayaji, A., & Yanco, H. (1994). The blind breeding the blind: adaptive behavior without looking. In *From animals to animats 3, Proceedings of the third international conference on simulation of adaptive behavior*. MIT Press.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, *55*, 189–208.

Tolman, E. C., & Honzik, C. H. (1930). ''Insight'' in rats. *University of California Publications in Psychology*, *4*, 215–232.

Touretzky, D. S., Wan, H. S., & Redish, A. D. (1994). Neural representations of space in rats and robots. In Zurada, J. M., Marks, R. J., & Robinson, C. J. (Eds.), *Computational intelligence: imitating life*. IEEE Press, pp. 57–68.

Trullier, O., Wiener, S., Berthoz, A., & Meyer, J. A. (1997). Biologically-based artificial navigation systems: review and prospects. *Progress in Neurobiology*, *51*, 483–544.

Von Wichert, G. (1998). Mobile robot localization using a self-organised visual environment representation. *Robotics and Autonomous Systems*, *25*, 185–194.

Yamauchi, B., & Beer, R. (1996). Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man,* *and Cybernetics-Part B*, *26*(3), 496–505, Special Issue on Learning Autonomous Robots.

Yamauchi, B., & Langley, P. (1997). Place recognition in dynamic environments. *Journal of Robotic Systems*, *14*(2), 107–120, Special Issue on Mobile Robots.

Yamauchi, B., Schultz, A., & Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *Proceedings of the IEEE international conference on robotics and automation (ICRA-98).*

Yamauchi, B., Schultz, A., & Adams, W. (1999). Integrating exploration and localization for mobile robots. *Adaptive Behavior*, *7*(2), 217–230.