Probabilistic Robot Navigation in Partially Observable Environments*

Reid Simmons and Sven Koenig
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213-3890
reids@cs.cmu.edu, skoenig@cs.cmu.edu

Abstract

Autonomous mobile robots need very reliable navigation capabilities in order to operate unattended for long periods of time. This paper reports on first results of a research program that uses partially observable Markov models to robustly track a robot's location in office environments and to direct its goal-oriented actions. The approach explicitly maintains a probability distribution over the possible locations of the robot, taking into account various sources of uncertainty, including approximate knowledge of the environment, and actuator and sensor uncertainty. A novel feature of our approach is its integration of topological map information with approximate metric information. We demonstrate the robustness of this approach in controlling an actual indoor mobile robot navigating corridors.

1 Introduction

We are interested in the task of long-term autonomous navigation in an office environment (with corridors, foyers, and rooms). While the state of the art in autonomous office navigation is fairly advanced, it is not generally good enough to permit robots to traverse corridors for long periods of time without getting lost. Evidence for this can be seen in recent AAAI-sponsored robot competitions [Konolige, 1994; Simmons, 1995], where the robots often got confused as to where they were, and had difficulty relocalizing once that occurred.

We contend that navigation can be made more reliable by having the robot explicitly represent spatial and sensor uncertainty. To this end, we have developed a navigation technique that uses Markov models to robustly track the robot's position and direct its course. A partially observable Markov decision process (POMDP) model is constructed from topological information about the connectivity of the environment, approximate distance information, plus sensor and actuator characteristics. The Markov model estimates the position of the robot in the form of probability distributions. The probabilities are updated when the robot reports that it has moved or turned, and when it observes features such as walls and corridor junctions. To direct the robot's behavior, a planner associates a directive (e.g., turn or stop) with every Markov state. Whenever the probability distribution of the Markov model is updated, the total probability mass for each directive is calculated, and the robot executes the one with the largest probability mass.

Our approach has several features that make it well-suited for the office navigation task. It explicitly accounts for uncertainty in actuation, sensor data and their interpretation, and the robot's position. It can utilize all available sensor information to track position, and is particularly amenable to adding new sources of sensor information. It seamlessly combines topological and metric map information, enabling the robot to utilize as much, or as little, metric information as it has available. It is also very reactive – once the robot believes it has strayed from the nominal (optimal) path, it will automatically execute corrective actions. On the other hand, it is relatively immune to temporary uncertainty in position. For example, even if the robot does not know for certain which of two parallel corridors it is traversing, it does not stop and replan, as long as the control directives associated with both corridors are the same. In this way, it can continue making progress towards its desired goal, while at the same time collecting sensor readings to help disambiguate its true location.

An important aspect of this work is that it must run in real time on board an actual robot. Problems include not only how to model the navigation problem as a POMDP, but also how to deal with memory and time constraints. While still preliminary, our experimental results, both in simulation and on the actual robot, are encouraging. In particular, they indicate that the approach produces very robust navigation, even when using estimates of the actual sensor and action models. While, to date, we have concentrated more on implementation and validation aspects of the approach, our work opens up new application areas for more theoretical results

*This research was supported in part by NASA under contract NAGW-1175 and by the Wright Laboratory and ARPA under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA, the Wright Laboratory, or the United States government.

in the area of planning with Markov models, including some of our own group's work [Chrisman, 1992; Goodwin, 1994; Koenig and Simmons, 1994].

2 Related Work

Most recent work in robotic office navigation has used a *landmark-based* approach that relies on topological maps whose nodes correspond to landmarks (locally distinctive places), such as corridor junctions, and whose edges indicate how the robot should navigate between nodes [Kortenkamp and Weymouth, 1994; Kuipers and Byun, 1988]. This approach is attractive because it does not depend on geometric accuracy and is reactive to sensed features of the environment (the landmarks). It suffers, however, from problems of sensors occasionally not detecting landmarks and of sensor *aliasing* (not being able to distinguish between similar landmarks). On the other hand, using purely metric maps is vulnerable to inaccuracies in both the map making and dead-reckoning abilities of the robot. While some researchers augment topological maps with approximate metric information, such information is primarily used to resolve topological ambiguities [Kuipers and Byun, 1988; Mataric, 1991; Simmons, 1994]. In contrast, our Markov model approach seamlessly integrates topological, landmark-based, information and approximate metric information.

Some navigation techniques represent uncertainty in position using models that presume a certain probability distribution, typically Gaussian [Kosake and Kak, 1992; Smith and Cheeseman, 1986]. While such models are efficient to encode and update, they are not ideally suited for office navigation. In particular, due to sensor aliasing, one often wants to encode the belief that the robot might be in one of a number of non-contiguous locations. This cannot be represented precisely using Gaussian distributions, but is quite easy for our Markov models. On the other hand, we need to tessellate space into discrete states, rather than representing position using real numbers. Thus, there is a tradeoff between the precision and expressiveness of the different models. We contend that for office navigation, however, that the added expressiveness of the Markov models outweighs the decrease in precision from discretization.

Like our own work, several researchers have investigated Bayesian approaches for probabilistic planning and execution monitoring in office navigation. [Nourbakhsh *et al.*, 1995] use a partially observable Markov model approach similar to ours, but do not utilize any metric information. The states of the robot are either at a topological node, or somewhere in a connecting corridor. In contrast, our approach can use estimates of how far the robot has traveled and sensor reports that occur within a corridor to further constrain the robot's location. For example, knowing that two corridor junctions are approximately 5 meters apart enables the robot to estimate when it is in the vicinity of the second junction, even if it misses seeing the junction.

Most of the other Bayesian approaches rely on metric maps. [Kirman *et al.*, 1991] and [Nicholson and Brady, 1994] use approaches based on temporal belief networks. With such methods the size of the models grows linearly with the amount of temporal lookahead, which limits their use to rather small lookaheads. [Dean *et al.*, 1993] use robot navigation as an example to describe a planning and monitoring algorithm that uses a totally observable Markov model, which assumes that the location of the robot is always known precisely. [Hu and Brady, 1994] use Bayesian techniques to detect unforeseen obstacles in an otherwise completely known environments.

3 Constructing the Markov Models

Before describing how we construct Markov models of an office environment, we introduce some terminology. A finite Markov model consists of a finite set of states S, a finite set of actions A, a set of actions $A(s) \subseteq A$ for each state $s \in S$, that can be executed in that state, and transition probabilities $p(s'|s,a)$ for all $s, s' \in S$ and $a \in A(s)$ (the probability that the new state is s' if action a is executed in state s). We also define a set of sensors $i \in I$. The sensors are characterized by observation probabilities $p_i(o|s)$ for all $s \in S$ and $o \in O(i)$ (the probability that sensor i reports feature o when the robot is in state s). Note that Markov models assume that the transition and observation probabilities are determined only by the current state of the robot (the "Markov assumption").

In our case, the Markov model is partially observable because the robot may never know exactly which state it is in. Instead, it maintains a belief of its current state in form of a probability distribution $p(s)$ over the states $s \in S$. The probability distribution is updated in two ways: When an action report a is received, indicating a move or turn, the new probabilities become:

$$p_{posterior}(s) = K \times \sum_{s' \in S \,|\, a \in A(s')} p(s|s',a) \times p_{prior}(s')$$

where K is a normalization factor to ensure that the probabilities all sum to one (this is necessary because not all actions are defined for all states). When a sensor report o is received from sensor i, indicating that a feature has been detected, the probabilities become:

$$p_{posterior}(s) = K \times p_i(o|s) \times p_{prior}(s)$$

The Markov model is constructed from three sources of information: the topology of the environment (which we presume can be easily obtained), general knowledge about office environments (such as that corridors are straight and perpendicular to each other), and approximate metric knowledge (obtained either from rough measurements or from general knowledge, such as the fact that, in our building, corridors are two meters wide and all doorways are between two and ten meters apart).

The map information is initially encoded as a graph of nodes and edges (Figure 1). A node represents a junction between corridors (and/or doorways or foyers). Nodes are connected by a pair of directed edges, which are augmented

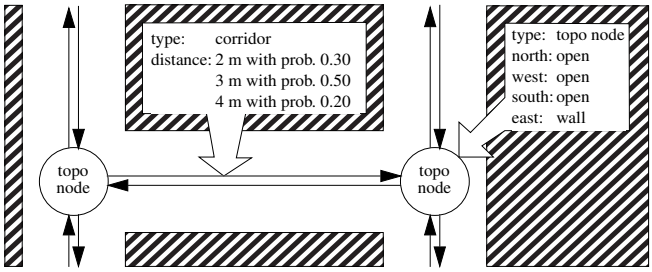


Figure 1: Augmented Topological Map

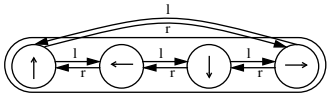


Figure 2: Group of Four Markov States

with approximate length information in the form of a probability distribution over possible lengths. Rooms and foyers (not shown) are also represented in the map.

The rest of this section describes how the augmented topological map is compiled into a Markov model.

Modeling Locations

Each Markov state encodes both the orientation and location of the robot. To insulate the model from low-level control aspects (such as turning to avoid obstacles), we encode the commanded heading of the robot rather than its instantaneous orientation. Since our corridors are straight and perpendicular to each other, it is sufficient to discretize orientation into the four compass directions: North, South, East, West. The spatial locations of the robot are also discretized. While more fine-grained discretizations yield more precise models, they also result in more memory requirements and more time-consuming computations. We use a resolution of one meter, which we have found to be sufficient.

Since our Markov states encode both orientation and location, four states are needed to fully represent each spatial location. Three actions are modeled: turning right 90 degrees (r), turning left 90 degrees (l), and going forward one meter (f). Right and left turn actions are defined for every state (Figure 2). Since they correspond to changes in commanded heading and not to changes in position, we have found it sufficient to model them deterministically. Some states also have "forward" actions defined for transitioning from location to location (note that forward actions are not defined for states that face walls). Dead-reckoning uncertainty is modeled by a self-transition, that is, the forward action transitions with some probability from a state into itself.

Modeling Corridors

Our representation of topological edges is a key to our approach. If the edge lengths are known exactly, it is simple to model the ability to traverse a corridor with a Markov chain that has forward actions between those states whose orientations are parallel to the corridor axis (Figure 3a). The model

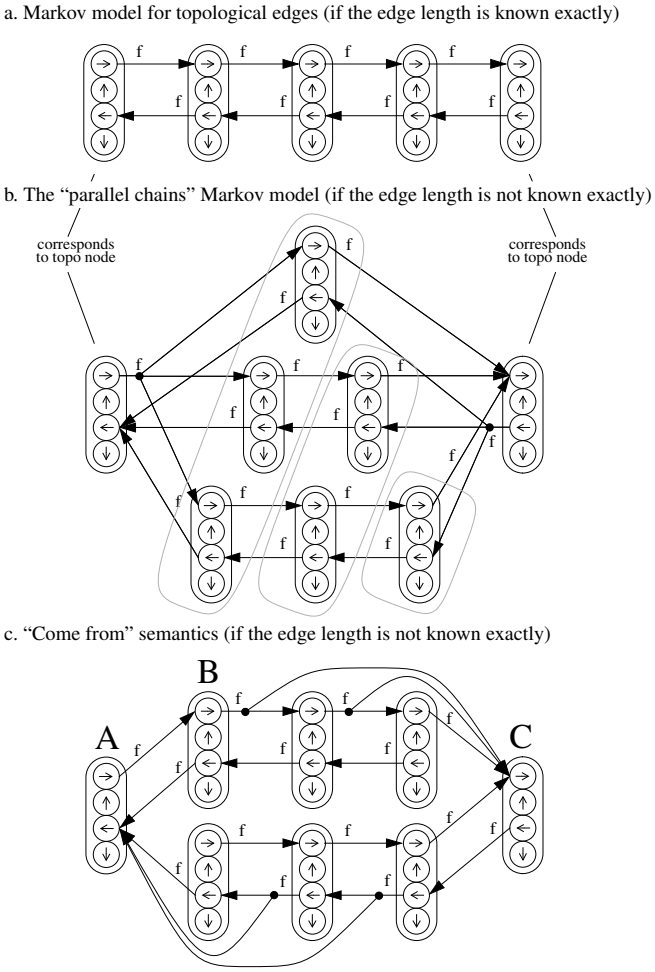
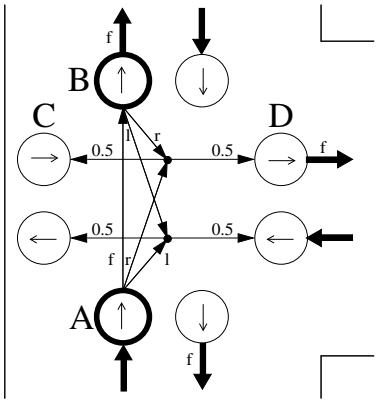


Figure 3: Representations of Topological Edges

becomes more complex when only approximate edge lengths are known. While one approach is to represent a corridor edge by a single Markov state [Nourbakhsh *et al.*, 1995], this loses the ability to utilize dead-reckoned information in doing position estimation.

Another approach is to model an edge as a set of parallel Markov chains, each corresponding to one of the possible lengths of the edge (Figure 3b). The transition probabilities into the first state of each chain are the same as the probability distribution over edge lengths associated with the topological map (see Figure 1). Each forward transition after that is deterministic (modulo dead-reckoning uncertainty — note that the identity transitions are not shown in these figures). While this representation best captures the actual structure of the environment, it is relatively inefficient: the number of states is quadratic in the maximum length of the edges.

As a compromise between fidelity and efficiency, our current implementation models edges by collapsing the parallel chains in a way that we call the "come from" semantics (Figure 3c). In this representation, the spatial location of a Markov state is known relative to the topological node from which the robot comes, but its location relative to the end of



(for clarity, only actions from the highlighted nodes are shown)

Figure 4: Representation of Corridor Junctions

the chain is uncertain (e.g., state B is 1 meter from A, but is between 1 and 3 meters from C). For each state, the forward transition probabilities are derived from the edge length probability distributions. When edge length uncertainty is large, the "come from" semantics can save significant space over the "parallel chains" representation. For example, for an edge between 2 and 10 meters long, the "come from" semantics needs only 80 states to encode the edge, compared to 188 for the "parallel chains."

Each edge in the "come from" semantics is actually represented using two chains, one for each of the corridor directions. Thus, if the robot travels some distance and then turns around, the model limits the positional uncertainty as the robot travels back to the last topological node. This is particularly useful when the robot realizes it has missed a junction, and turns around to head back.

Modeling Junctions and Doorways

Unfortunately, we cannot represent corridor junctions simply with a single group of four Markov states, since the spatial resolution of a Markov state is one meter, but our corridors are two meters wide.

While one approach would be to represent junctions using four (two by two) groups of four Markov states each, we achieve nearly the same result with four groups of two states each, which both saves space and makes the model simpler (Figure 4). The basic idea is that turns within a junction are non-deterministic, with equal probability of transitioning to one of the two states of the appropriate orientation in the junction. For example, in entering the junction of Figure 4 from the South, the robot would first encounter state A, then state B if it continues to move forward. If it then turns right, it would be facing East, and would transition to either states C or D with equal probability. This models agrees with how the robot actually behaves in junctions. In particular, it captures the uncertainty that arises due to the fact that the robot turns with a non-zero turn radius.

Doorways can be modeled much more simply, since the width of our doors is approximately the resolution of the Markov model. A single exact-length edge (Figure 3a) leads

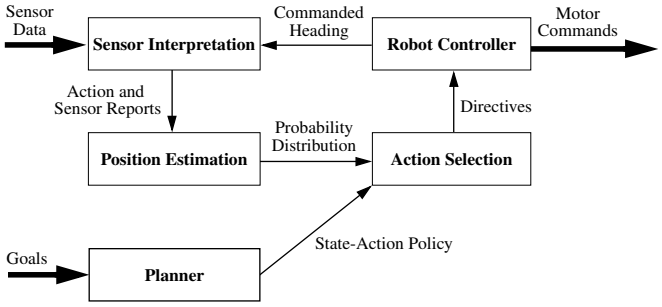


Figure 5: Navigation System Architecture

through a door into a room. Similarly to [Nourbakhsh *et al.*, 1995], doorway edges have an associated probability p that the door is open. Then, the observation probabilities associated with seeing a doorway are:

$$p_i(o|\text{door}) = p \times p_i(o|\text{open-door}) + (1-p) \times p_i(o|\text{closed-door})$$

Modeling Foyers and Rooms

We are developing adequate models for large open spaces (foyers and rooms). Currently, we tessellate a foyer into a matrix of locations. From each location, the forward action has some probability of transitioning straight ahead, but also some probability of self-transitioning and moving to diagonally adjacent states. While this model corresponds well with our observations about how the robot actually performs in such spaces, it is deficient in that it requires the exact length and width of the foyer. Although this model could also be used to represent rooms, it is probably overly complex for that purpose: we are currently leaning towards representing rooms using a single group of four states, each of which has a high probability of self-transitioning.

4 The Navigation System Architecture

The overall system architecture consists of five main components (Figure 5). The robot controller performs local obstacle avoidance while trying to travel along a commanded heading. The sensor interpretation component converts raw data from the wheel encoders and sonar into higher-level action reports (heading changes and distance traveled) and sensor reports (features detected). Position estimation uses these reports and the Markov model to maintain a belief about the current location of the robot. Action selection uses this probability distribution, along with a goal-directed policy produced by the planner, to choose directives which are sent to the controller to change the robot's heading or make it stop. These directives are also fed back to sensor interpretation, since interpretation of features is often heading specific.

To date, the work reported here has focused on position estimation and action selection. The robot controller and sensor interpretation components are essentially the same as those used in our previous work in landmark-based navigation [Simmons, 1994], and we have not yet put significant effort into the planner.

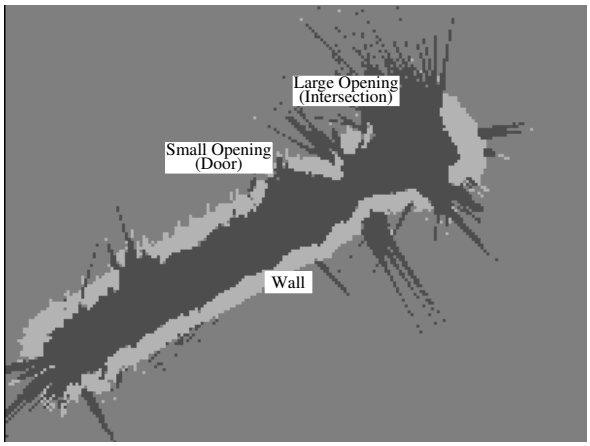


Figure 6: Occupancy Grid with Corridor Features

Robot Controller

The main task of the robot controller is to head in a given direction while avoiding obstacles. To do that, it uses a potential field approach [Arkin, 1987], in which obstacles are represented as repulsive forces and the desired heading is an attractive force. The robot sums the force vectors and locally moves in that direction, modulating its speed if necessary to avoid collisions.

The directives supplied to the controller are to make it stop, go, and change heading. While the Markov model represents turns and moves as discrete actions, in reality the robot does not stop to turn, but continually moves forward, even while turning. In addition, heading changes are cumulative, so that two successive right turn directives, for instance, results in a smooth 180 degree turn.

Sensor Interpretation

The task of the sensor interpretation component is to convert the continual motion of the robot into discrete action reports, and to produce sensor reports from the raw sensor data that indicate the observation of high-level features, such as walls and corridor openings.

The sensor interpretation component periodically receives reports from the robot's *dead-reckoning*, which uses internal sensors (wheel encoders) to estimate position and orientation. This information is combined with the robot's commanded heading to produce a "virtual odometer" that keeps track of the distance traveled along that heading. This is needed so that the distance the robot travels in avoiding obstacles is not counted in determining how far it has traveled along a corridor. After each meter of cumulative travel, the sensor interpretation reports that one forward action has occurred. Similarly, the robot controller reports when its commanded heading has changed, and this is reported (in units of 90 degree turns) to the position estimation component.

Sonar readings are bundled into three "virtual sensors" that report observations of walls and openings of various sizes (small, medium and large) in front of the robot and to its immediate left and right. An occupancy grid [Elfes, 1989], which probabilistically combines sonar sensor readings taken over time as the robot travels, is used to filter noisy sensor readings and produce a more global view of the robot's surroundings (Figure 6). The occupancy grid is processed by projecting a sequence of rays perpendicular to the robot's commanded heading (thus, it is independent of the robot's actual orientation), until they intersect an occupied grid cell. The rays are then analyzed geometrically. If the end points of the rays can be fit to a line reasonably well (i.e., with a small chi-squared statistic), then a wall has been detected with high probability. An opening is indicated by a contiguous sequence of long rays.

Position Estimation

The virtual sensor and action reports are used to update the probability distribution over the Markov states according to the update rules shown in Section 3. These rules need the transition probabilities for actions $p(s'|s,a)$ and the observation probabilities for virtual sensors $p_i(o|s)$. The transition probabilities are derived from edge length distributions in the map plus knowledge of dead-reckoning uncertainty. The observation probabilities must be estimated or learned. To simplify the problem, rather than characterizing each individual state, we characterize classes of states, such as `wall`, `open` (corridor junctions), `closed-door`, and `open-door`. Then, we create a table containing feature/state class pairs that encode the probability that the sensor reports a given feature when the robot is next to that particular class of states. For example, the left virtual sensor is partially characterized by:

p(`wall`|`open`)	= 0.05
p(`small_opening`|`open`)	= 0.20
p(`medium_opening`|`open`)	= 0.40
p(`large_opening`|`open`)	= 0.30
p(`nothing`|`open`)	= 0.05

These probabilities indicate that junctions are most commonly detected as medium-sized openings, but can often be seen as large or small openings (although they are hardly ever confused for walls). The observation probabilities of the feature `nothing`, which is used to indicate that a sensor has made no determination, are chosen so that, if the sensors reports `nothing`, the overall probability distribution is unaffected. While these values represent our best guesses, we have implemented learning algorithms to determine action transition and observation probabilities more precisely.

In general, forward actions tend to increase positional uncertainty, due to non-deterministic transitions, while observations tend to decrease it. In certain cases, however, the effects of a forward action can dramatically decrease uncertainty. This occurs when there is some probability that the robot is in states with no forward actions ($f \notin A(s)$). Such states are prevalent — for instance, all states within a corridor whose orientation is perpendicular to the axis of the corridor (see Figure 3). In practice, this effect can be seen when the robot turns at an intersection: Before the turn, there is often some probability that the robot has not yet reached the intersection. After the robot has turned and successfully moved

forward a bit, the probability that it is still in the original corridor drops to zero. We believe this is a major factor in keeping the positional uncertainty low, even when the robot travels long distances.

When incorporating sensor reports, care must be taken to preserve the Markov assumption. Since reports by the same sensor at the same location are not independent (since they depend on the same occupancy grid cells), multiple reports cannot be aggregated. Instead, we retract the old sensor report before updating with the new report, which can be done easily as long as no action updates occur between the two reports.

Action Selection

To control the robot's goal-directed behavior, our planner (see below) associates a directive $d(s) \in D$ with each Markov state (note: these should not be confused with the set of actions $A(s)$ defined for the Markov model). The four directives are: change heading by 90 degrees (turn right), -90 degrees (turn left), 0 degrees (go forward), and stop. The action selection component chooses new directives based on the probability distribution of the Markov model.

A straightforward strategy is to choose the directive associated with the state s that has the highest probability [Nourbakhsh *et al.*, 1995]. While this strategy may be adequate when each topological entity is associated with a single Markov state, it does not work well in our models. For example, since corridor junctions are modeled using several states for each orientation, it is reasonable to consider all of their recommendations when deciding which directive to issue.

A selection strategy with this property is the "best-action" strategy, in which the *probability mass* of each directive is calculated and the one with the highest total probability is chosen:

$$\arg\max_{d \in D} \sum_{s \in S | d(s)=d} p(s)$$

A variation on this is the "best-above-threshold" selection strategy, which chooses the best directive only if its probability mass is above some threshold, otherwise the current directive remains in effect. We have investigated this strategy because we thought it would reduce the chances of making wrong moves due to spurious false positive sensor reports. Experimental evidence, however, both in simulation and with the real robot, indicate that the "best-action" strategy is in fact superior in reducing the number of erroneous moves.

Reinforcement learning researchers, such as [Chrisman, 1992; Tenenberg *et al.*, 1992], often use other voting schemes, such as the following: Let $gd(s, d)$ be the shortest distance from state $s \in S$ to the goal if the robot executes directive $d \in D(s)$ and then behaves optimally. The strategy chooses the directive with the smallest expected goal distance:

$$\arg\min_{d \in D} \sum_{s \in S} p(s) gd(s, d)$$

This scheme allows one, for example, to choose the second best action if all states agree on the second best action, but disagree on the best action. While this scheme is attractive, we did not implement it because it would require substantial changes to our path planner.

Planning

While opportunities abound for applying probabilistic planning techniques to this problem, we currently use a very simple symbolic path planner, a variant on the one used for our landmark-based navigation.

The planner uses A* search in the augmented topological map to find a path to the goal. It uses this plan skeleton to assign preferred headings to the edges and nodes in the map, based on the expected total travel distance to the goal and estimates about how long it takes to turn. Directives are then associated with the Markov states: a "go forward" directive is assigned to each state whose orientation is the same as the preferred heading of its associated topological entity. The remaining states are assigned actions that will turn the robot towards the desired heading. Finally, a "stop" directive is assigned to the goal state and to nearby states (which helps to increase the total probability mass of the "stop" directive when the robot reaches the goal).

Our planner, and the voting heuristics used in action selection, are clearly inferior compared to optimal POMDP solutions (in which directives are assigned to probability distributions, rather than individual states). For example, unlike POMDP algorithms, our planner cannot decide to take actions whose only purpose is to gather information. Sometimes, however, it can be advantageous to gather additional information that helps the robot to reduce positional uncertainty, even if that requires it to move away from the goal temporarily.

At present, however, it is infeasible to determine even approximate POMDP solutions given the size of our state spaces and our real-time constraints [Lovejoy, 1991]. [Cassandra *et al.*, 1994], for instance, report that their POMDP method can solve a problem with 23 states in under half an hour, while the model for just half of one floor of our building has over 3000 states. We still intend to explore POMDP algorithms, however, given recent advances in approximate algorithms [Parr and Russell, 1995] and the hope that the restricted topology of our Markov models might make them more amenable to efficient solutions.

5 Experiments

While Markov models are expressive and relatively efficient, they make strong independence assumptions. Empirical evidence is needed to determine whether, in this case, the Markov assumption is satisfied well enough to yield good, reliable navigation performance. In this section, we report on experiments in two environments for which the Markov assumption is only an approximation: a realistic simulation of a prototypical office corridor environment, and an actual mobile robot navigating in our building. The same navigation code is used for both sets of experiments, since the simulator and the robot have the exact same interfaces.

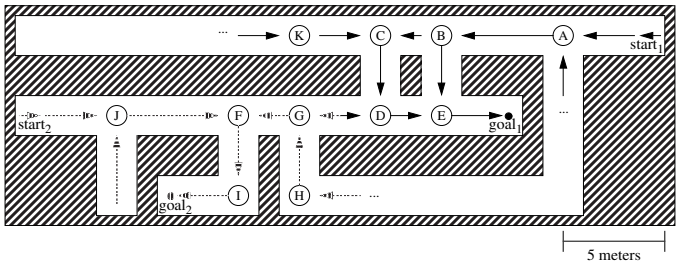


Figure 7: An Office Corridor Environment

	best-action			best-above-threshold		
path	freq.	time s	speed cm/s	freq.	time s	speed cm/s
ABE	12	68.2	25.7	5	63.6	27.5
ABCDE	3	79.7	29.5	8	81.0	29.0
ABCKCDE	—	—	—	2	104.1	N/A

Table 1: Experiment 1

	best-action			best-above-threshold		
path	freq.	time s	speed cm/s	freq.	time s	speed cm/s
JFI	11	60.6	28.9	8	65.4	26.8
JFGFI	2	91.5	25.7	—	—	—
JFGHGFI	1	116.0	23.7	5	120.2	22.9
JFGFGFI	1	133.0	22.2	—	—	—
JFGDGFI	—	—	—	2	176.5	N/A

Table 2: Experiment 2

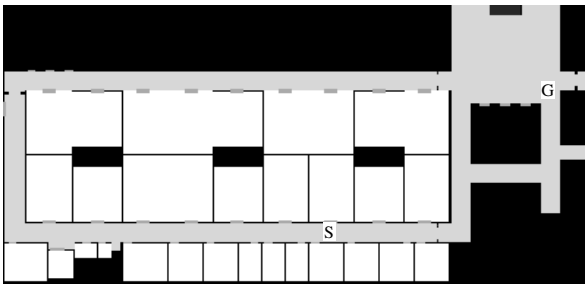


Figure 8: Wean Hall at CMU

5.1 Experiments with the Simulator

Two navigation experiments were performed with the robot simulator in the corridor environment shown in Figure 7. The topological map has 17 nodes and 36 directed edges. We modeled the uncertainty of the length of a topological edge as a uniform distribution over the interval ranging from 80 to 150 percent of the real length of the edge. The resulting Markov model has 1184 Markov states. The initial positional uncertainty for both experiments is minimal: the initial probability for the robot's actual location is about 90 percent. The remaining probability mass is distributed in the vicinity of the actual location.

In the first experiment, the task was to navigate from $start_1$ to $goal_1$. The preferred headings assigned by our planner are shown with solid arrows. Note that the preferred heading between B and C is towards C because, even though the goal distance is a bit longer, this way the robot does not have to turn around if it overshoots B. We ran a total of 15 trials for both the best-action and the best-above-threshold strategies, all of which were completed successfully (Table 1).

The robot has to travel a rather long distance from $start_1$ before its first turn. Since this distance is uncertain and corridor openings are occasionally missed, the robot occasionally overshoots B, and then becomes uncertain whether it is really at C or B. However, since the same directive is assigned to both nodes, this ambiguity does not need to be resolved: the robot turns left in both cases and then goes straight. The same thing happens when it gets to D, since it thinks it may be at either D or E. The robot eventually corrects its beliefs when, after turning left and traveling forward, and it detects an opening to its left. At this point, the robot becomes fairly certain that it is at E. A purely landmark-based navigation technique can easily get confused in this situation, since it has no expectations about seeing this opening, and can only attribute it to sensor error (which, in this case, is incorrect).

In the second experiment, the robot had to navigate from $start_2$ to $goal_2$. The preferred headings for this task are shown with dashed arrows. Again, we ran 15 trials for both action selection strategies (Table 2).

For reasons that are similar to those in the first experiment, the robot can confuse G with F. If it is at G but thinks it is probably at F, it turns right and goes forward. However, when it detects the end of the corridor but does not detect a right corridor opening, it realizes that it must be at H rather than I. Since the probability mass has now shifted, it turns around and goes over G, F, and I to the goal. This shows that our navigation technique can gracefully recover from misjudgements based on wrong sensor reports – even if it takes some time to correct its beliefs. It is important to realize that this behavior is not triggered by any explicit exception mechanism, but results automatically from the way the position estimation and action selection interact.

5.2 Experiments with Xavier

Xavier, our indoor mobile robot, is built on an RWI B24 base and includes bump sensors, sonars, a laser range sensor, and a color camera on a pan-tilt head. Control, perception and planning are all carried out on two on-board, multi-processing 486-based machines.

As mentioned, the probabilistic navigation system uses a modified version of the planner and essentially the same robot controller and sensor interpretation components as our landmark-based navigation system. Thus, differences in performance can be directly attributed to the different navigation approaches. In addition, to facilitate comparisons we ran Xavier along the same routes as reported in [Simmons, 1994]. In particular, the robot traversed from point S to G and back again (Figure 8) in some trials (45 meters each way), and in some circumnavigated around the building (150 meters).

The topological map used to represent the corridors in Figure 8 has 95 nodes and 180 directed edges. As with the simulator trials, the edge lengths ranged uniformly from 80 to

150 percent of the real corridor length. The resulting Markov model has 3348 states.

In 25 trials (mostly back and forth between points S and G), the robot successfully reached its goal in 22 cases, averaging 30 cm/s while traversing a total of over a kilometer. In two of those cases, the robot missed seeing a junction, but turned back when it realized it had probably gone too far, and successfully continued. This success rate of 88% compares favorably with the 80% rate reported in [Simmons, 1994].

The main difference is that the probabilistic navigation scheme uses all available sensor information to help localize itself. For example, while the probabilistic navigation uses the robot's dead-reckoning to directly constrain its position estimates, the landmark-based navigation uses metric information in only two ways: it ignores landmarks that are reported before a minimum distance has been traveled, and turns around after a maximum distance. Similarly, the probabilistic navigation scheme utilizes all sensor reports, while the landmark-based scheme pays attention only to those features that might correspond to the expected landmark. One effect of this is that the probabilistic navigation scheme tends to turn the robot earlier when entering a junction: it often gets enough confidence from a single sensor report, while the landmark-based scheme needs several (e.g., seeing both an opening to the side and the end of the corridor ahead) before it decides to turn.

The few remaining failures are attributable to two sources: Occasionally the action selection heuristics enter a limit cycle and continually turn the robot (we suspect this is due to a software bug). More fundamental is that the local obstacle avoidance will, especially in foyers, move the robot a significant distance orthogonally to its commanded heading. Since this is not currently reported, the robot's position estimation becomes very inaccurate. We can remedy this by reporting side motions and adding a "slide" action to the Markov model that will cause the appropriate state transitions.

6 Future Work and Conclusions

This paper has presented our first efforts at using partially observable Markov models (POMDPs) for autonomous office navigation. The approach enables a robot to utilize all its sensor information, both positional and feature-based, in order to robustly track its location. A simple path planner and action selection heuristics are used to direct the robot's goal heading. Advantages of this approach include the ability to account for uncertainty in the robot's initial position, actuator uncertainty, sensor noise, and uncertainty in the interpretation of the sensor data. Also, by integrating topological and metric information, the approach easily deals with uncertainty arising from incomplete descriptions of the environment.

We are extending this work in several directions. We have implemented methods, based on EM learning techniques, that passively refine metric map information as well as the sensor and action models, and will be testing it with Xavier. In addition, we are developing improved learning techniques that are more resistant to violations of the Markov assumption. We intend to pursue planning and action selection algorithms that approximate optimal POMDP policies, and to compare their performance to the greedy heuristics described here. Finally, we intend to add new sources of sensor information, primarily vision-based feature detectors.

The implemented probabilistic navigation system has demonstrated its reliability, both in simulation and on Xavier, even in the face of significant uncertainty. We believe that such probabilistic navigation techniques hold great promise for getting robots reliable enough to operate unattended for long periods of time in complex, uncertain environments.

Acknowledgements

Thanks to Lonnie Chrisman, Richard Goodwin and Joseph O'Sullivan for helping to implement parts of the navigation system and for many valuable discussions. Swantje Willms helped perform some of the experiments with the simulator.

References

[Arkin, 1987] R.C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 264–271, 1987.

[Cassandra *et al.*, 1994] A.R. Cassandra, L.P. Kaelbling, and M.L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the AAAI*, pages 1023–1028, 1994.

[Chrisman, 1992] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distanctions approach. In *Proceedings of the AAAI*, pages 183–188, 1992.

[Dean *et al.*, 1993] T. Dean, L.P. Kaelbling, J. Kirman, and A. Nicholson. Planning with deadlines in stochastic domains. In *Proceedings of the AAAI*, pages 574–579, 1993.

[Elfes, 1989] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, pages 46–57, 6 1989.

[Goodwin, 1994] R. Goodwin. Reasoning about what to plan. In *Proceedings of the AAAI*, page 1450, 1994.

[Hu and Brady, 1994] H. Hu and J.M. Brady. A Bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots*, 1(1):69–92, 1994.

[Kirman *et al.*, 1991] J. Kirman, K. Basye, and T. Dean. Sensor abstractions for control of navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2812–2817, 1991.

[Koenig and Simmons, 1994] S. Koenig and R. Simmons. How to make reactive planners risk-sensitive. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 293–298, 1994.

[Konolige, 1994] K. Konolige. Designing the 1993 robot competition. *AI Magazine*, 15(1):57–62, 1994.

[Kortenkamp and Weymouth, 1994] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the AAAI*, pages 979–984, 1994.

[Kosake and Kak, 1992] A. Kosake and A. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 2177–2186, 1992.

[Kuipers and Byun, 1988] B.J. Kuipers and Y.-T. Byun. A robust, qualitative method for robot spatial learning. In *Proceedings of the AAAI*, pages 774–779, 1988.

[Lovejoy, 1991] W.S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991.

[Mataric, 1991] M.J. Mataric. A distributed model of mobile robot environment-learning and navigation. Technical Report TR–1228, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991.

[Nicholson and Brady, 1994] A.E. Nicholson and J.M. Brady. Dynamic belief networks for discrete monitoring. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1593–1610, 1994.

[Nourbakhsh *et al.*, 1995] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish: An office-navigating robot. *AI Magazine*, 16(2):53–60, 1995.

[Parr and Russell, 1995] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the IJCAI*, pages 1088–1094, 1995.

[Simmons, 1994] R. Simmons. Becoming increasingly reliable. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 152–157, 1994.

[Simmons, 1995] R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 1995. (in press).

[Smith and Cheeseman, 1986] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5:56–68, 1986.

[Tenenberg *et al.*, 1992] J. Tenenberg, J. Karlsson, and S. Whitehead. Learning via task decomposition. In *Proceedings of the Conference "From Animals to Animats"*, pages 337–343, 1992.