Probabilistic Navigation in Partially Observable Environments

Reid Simmons and Sven Koenig

School of Computer Science, Carnegie Mellon University

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. We have developed an approach that uses partially observable Markov models to robustly track a robot's location and integrates it with a planning and execution monitoring approach that uses this information to control the robot's 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, actuator uncertainty, and sensor noise. A novel feature of our approach is its integration of topological map information with approximate metric information. We demonstrate the reliability of this approach, especially its ability to smoothly recover from errors in sensing.

1. Introduction

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 [6, 15], where the robots often got confused as to where they were, and had difficulty relocalizing once that occurred. While other aspects of the overall navigation problem, such as path planning, are important, in practice, the major difficulty in achieving reliable navigation is in maintaining a good estimate of the robot's whereabouts.

We have developed a navigation technique that uses Markov models to robustly track a robot's position and orientation and to control its goal-oriented actions. The approach has several features that make it well-suited for the office navigation task. For one, it explicitly accounts for uncertainty in sensors, actuation, and the robot's initial position. It seamlessly combines topological and metric map information, enabling the robot to utilize as much, or as little, *a priori* metric information as it has available. The approach can utilize all of the robot's sensor information to track position, and is particularly amenable to adding new sources of sensor information.

We presume that the robot can be easily supplied with topological information about the connectivity of the environment (junctions between corridors, doorways, and foyers). In addition, we presume that some prior metric information may be available – for instance, we may know that all intersections are at 90 degree angles, that corridors are two meters wide, and that the distance between doorways is no more than ten meters. This information is used to construct a topological map, with the nodes representing junctions and the edges augmented with approximate length information. The augmented topological map is then compiled into a partially observable Markov chain.

The Markov chain maintains the position and orientation of the robot in the form of probability distributions over the states of the Markov chain. This information is updated using Bayes' rule as the robot moves forward, turns, and processes sensor information to detect distinctive features such as walls, doorways and corridor junctions. To control the robot's behavior, a planner associates an action with every Markov state. Whenever the probability distribution of the Markov chain is updated, the total probability mass for each action is calculated, and the action with the largest probability mass is chosen for execution.

An advantage of this approach is that it is very reactive – if the robot strays from the nominal (optimal) path, it will automatically execute corrective action once it realizes its mistake (that is, once there is enough probability that the robot is in a different part of the environment). The approach is also 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 will not have to stop and replan, as long as both corridors recommend the same action. In this way, the robot can continue making progress towards its desired goal, while at the same time collecting evidence, in the form of sensor readings, that can help to disambiguate its true location.

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 [7, 9, 11]. 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 different landmark locations). 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 the edges of the topological maps with approximate metric information, such as ranges of distances between nodes, such information is primarily used to resolve topological ambiguities, and is not integral to their navigation approach [9, 14]. In contrast, our Markov model approach utilizes both the topological, landmark-based information and approximate metric information to perform position estimation and control.

Like our own work, several researchers have investigated the use of a Bayesian approach for representing uncertainty in probabilistic path planning and execution monitoring in office navigation. [12] uses a similar approach that tracks position using a partially observable Markov model, but does not encode any metric information at all. The states of the robot are either at a topological node, or somewhere in a connecting corridor. A planner finds the best (deterministic) path to the goal, and replans if the currently most likely state is no longer on the planned path. In contrast, by incorporating metric information, our approach can utilize both knowledge how far the robot has traveled and sensor reports that occur within a corridor – for instance, seeing a clock in the middle of a corridor can place great constraints on the robot's location, even though it is not a signal that the robot is at a topologically interesting place. Our approach also enables the robot to continue making progress even in the face of significant positional uncertainty, without having to stop and replan.

Other Bayesian approaches to navigation include [5], in which the robot has to track a mobile target in a known building with small unforeseen obstacles, while localizing itself in order not to get lost. Temporal belief networks with a limited lookahead are used to derive which action to execute next, based on sensor reports that (probabilistically) differentiate locally distinct places. In similar work, [4] assumes that unforeseen obstacles might block the robot's path in an otherwise completely known environments. A decision-theoretic approach is used to interpret sonar sensor reports and to decide whether the current path is blocked. While these approaches are similar to ours in their Bayesian incorporation of probabilistic sensor information, they differ in their reliance on accurate metric maps.

Other researchers have represented uncertainty in position using models that presume a certain probability distribution, typically Gaussian [8, 16]. While such models are efficient to encode and update, they make assumptions that are not necessarily valid for the office navigation domain. 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 (but similar looking) locations. This cannot be represented precisely using Gaussian distributions, but is quite easy to do using 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 Gaussian and Markov representations. We contend, however, that for the task of office navigation, the added expressiveness of the Markov models outweighs the need to discretize the environment.

3. Using Markov Models for Probabilistic Navigation

The task being addressed is long-term autonomous navigation in a peopled office environment (with corridors, rooms, and foyers). The navigation problems consists of both ascertaining the robot's position and directing its course. Our approach deals with both these aspects: it enables the robot to reliably know where it is (and, failing that, to know when it is lost), and includes capabilities to guide the robot towards a given goal, even in the face of uncertainty about the robot's exact position in the environment.

Our basic approach is to use a partially observable Markov model to track the robot's position and to combine it with planning and execution monitoring that uses this information to control the robot's actions. A 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 conditional 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). In our case, the Markov model is *partially observable* because, after executing an action, the robot is not told what the new state is. Instead, it gets clues from sensors i whose reports O_i depend probabilistically on the state. The sensors are characterized by the conditional observation probabilities $p_i(o|s)$ for all $s \in S$ and $o \in O_i$.

Since the robot does not know for certain which state it is in, it maintains a belief of its current state in form of a probability distribution $p(s)$ over the states $s \in S$. Updating these probabilities is fairly straightforward using Bayes' rule. If the robot executes action a, the new probabilities are:

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

where K is a normalization factor to ensure that the probabilities all sum to one. If the robot receives sensor report o from sensor i, the probabilities become:

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

While planning in this framework could theoretically be done with POMDP (Partially Observable Markov Decision Process) algorithms [2], all known POMDP algorithms are intractable even for modest sized state spaces [10]. Our approach, therefore, is to have a planner associate a desired action $a_{pl}(s)$ with each Markov state. Since there is typically uncertainty over which state the robot is actually in, the total *probability mass* of each action is calculated and the action with the highest probability is executed:

$$\arg\max_{a \in A} \sum_{s \in S | a_{pl}(s) = a} p(s)$$

While Markov models are expressive and relatively efficient and straightforward to use, they make strong independence assumptions, for instance that the outcome of an action is independent of previously executed actions. Similarly, every sensor report conditioned on the current state is assumed to be independent of all previous reports from that sensor, and any others. While this Markov assumption is often a good approximation, one can not expect the world to be ideally Markovian. For example, repeatedly using the same sensor in the same location will usually produce highly correlated results. Our navigation approach employs several techniques to

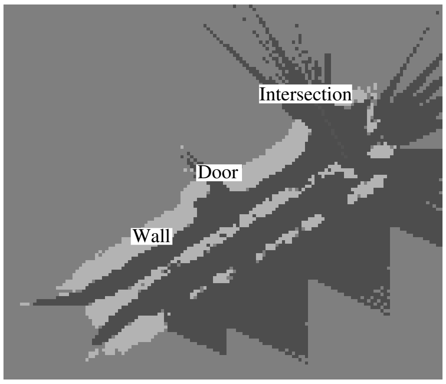

Figure 1: Evidence Grid Showing Features in the Corridor

help preserve the Markov assumption. For example, we bundle sets of features that are derived from the same sensor information (and, hence, are not independent) into "virtual sensors," and ensure that, for each virtual sensor, the Markov model utilizes only one sensor report at a time. In the end, however, one has to test empirically whether the Markov assumption is satisfied well enough for this approach to achieve good performance.

The Markov model must be integrated with the rest of the robot control and perception capabilities. For control purposes, note that we use the Markov model to supply a desired goal heading, not to control the low-level behavior of the robot. For that, we use a potential field approach [1] which combines goal-directed navigation with local obstacle avoidance. Obstacles are represented as repulsive forces and the goal heading is represented as an attractive force. The robot sums the force vectors and locally moves in that direction.

For perception, the Markov model needs to be supplied with both positional and feature-based information. Positional information comes from the robot's *dead-reckoning*, which uses internal sensors (wheel encoders) to estimate position and orientation. In particular, the robot maintains a "virtual odometer" that keeps track of the distance traveled along the commanded goal heading. This virtual odometer is needed so that the distance the robot travels in avoiding obstacles is not counted in determining how far it has traveled along the corridors.

For the feature-based capabilities, we process an evidence grid to detect walls and openings of various sizes. Using an evidence grid [3], which probabilistically combines sonar sensor readings taken over time as the robot travels, helps to filter noisy sensor readings and produces a more global view of the robot's surroundings (Figure 1). The evidence grid is processed by projecting rays perpendicular to the direction of travel, until they intersect an occupied grid cell. Geometric processing is then applied to the end points of the rays. For example, if the points 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.

As mentioned, sets of non-independent features are combined into virtual sensors.

Currently, we have virtual sensors for perceiving to the immediate left, right, and in front of the robot. The virtual sensors must be characterized by their conditional observation probabilities $p_i(o|s)$. Rather than characterizing each individual state s, we characterize classes of states. For instance, the states of the world to the left of the robot can be classified as "Wall," "Door," or "Open" (corridor junction) and the left virtual sensor can detect walls and small, medium and large openings. Then, a partial characterization of the left sensor is given by:

$$
\begin{aligned}
&p(\text{Wall}|\text{Open}) &&= 0.05\\
&p(\text{Small\_Opening}|\text{Open}) &&= 0.20\\
&p(\text{Medium\_Opening}|\text{Open}) &&= 0.40\\
&p(\text{Large\_Opening}|\text{Open}) &&= 0.30\\
&p(\text{Nothing}|\text{Open}) &&= 0.05
\end{aligned}
$$

which indicates that corridor junctions are usually detected as medium-sized openings, but can often be seen as large or small openings (although hardly ever are they confused for walls). The feature "Nothing" is used to indicate that the sensor has made no determination.

4. Constructing the Markov Model

Three sources of information are used to construct the Markov model: the topology of the environment, general domain knowledge about office environments, and approximate knowledge about the lengths of corridors. In this paper, we assume that the environment is stationary, and do not consider the problem of learning metric or topological map information.

The topological information is encoded in a map structure of nodes and edges (Figure 2). A topological node represents a junction between corridors and/or doorways. A pair of directed edges connects nodes. Part of our general domain knowledge includes the assumptions that corridors are straight and that junctions occur at 90 degree angles. Under these assumptions, it is sufficient to discretize orientation into the four compass directions: North, South, East, West (while the orientation of the robot may temporarily vary as it turns to avoid obstacles, its overall goal-directed heading can be safely assumed to be in one of those four directions).

Topological nodes are annotated with the type of environment (door, open, wall) that exists in each of the four directions. The edges are augmented with approximate length information, in the form of a probability distribution over possible lengths (Figure 2). Both the topological map and distance estimates are information that can be easily acquired by humans (and robots) by simply exploring the environment (as opposed to the difficulty and cost of obtaining exact distance information, since one has to carefully map the environment). Distance estimates can also be obtained from general information about the office environment. For example, we know that in our building corridors are two meters wide, and doorways are at least two meters apart but no more than ten meters apart.

The Markov model is constructed directly from the augmented topological map.

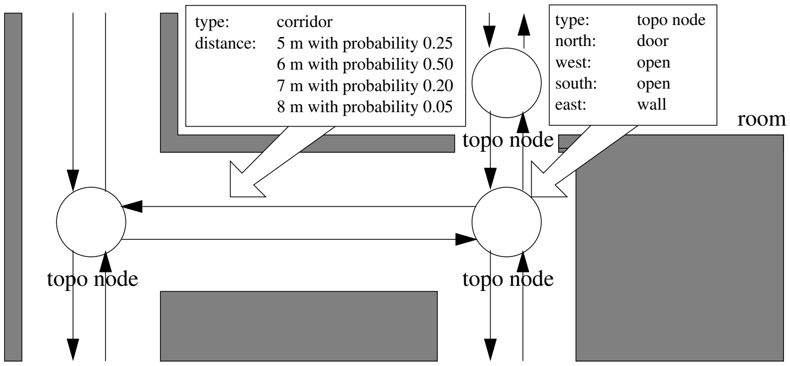

Figure 2: Augmented Topological Map

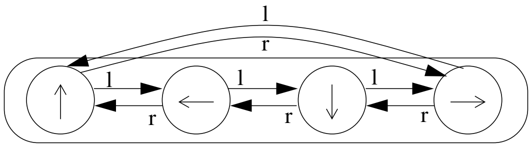

Figure 3: Group of Four Markov States With Turn Action Transitions

The spatial locations of the robot are discretized — the more fine-grained the discretization, the more precise the model, but also the more time-consuming the computations. We have found a resolution of one meter to be sufficient. Since each Markov state encodes both the orientation and location of the robot, we need four Markov states (corresponding to the four discrete orientations) to fully represent each spatial location.

Four actions are modeled: turning left 90 degrees, turning right 90 degrees, going forward 1 meter (the resolution of the model), and stopping. While, in practice, we have found it sufficient to model turns deterministically (Figure 3), it would be easy to make them probabilistic — for instance, having the left turn action change orientation 95% of the time, but 5% of the time leave the robot in the same Markov state.

Our representation of topological edges is a key to our approach. If the edge lengths are known exactly, it is simple to construct a Markov chain that models the ability to traverse a corridor, including turning around in the middle of the corridor to head back in the opposite direction (Figure 4a). The model becomes more complex when the edge length is known only approximately. While one approach is to represent a corridor edge by a single Markov state [12], this loses the ability to utilize dead-reckoned information in doing position estimation. In practice, this sort of information is very powerful, and we do not want to eliminate it from consideration.

Another approach is to model an edge as a series of Markov chains, each corresponding to one of the possible lengths of the edge (Figure 4b – for space reasons we show only the east-facing states). The transition probabilities into the first state

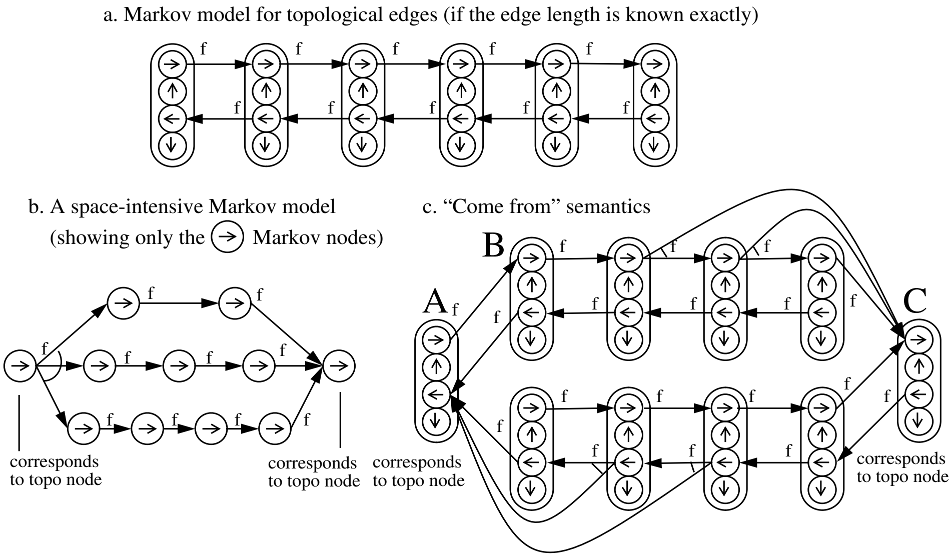

Figure 4: Representations of Topological Edges

of each chain would be determined by the probability distribution over edge lengths associated with the topological map. Each forward transition after that would be deterministic (except for dead-reckoning uncertainty, as described below). While this representation best captures the actual structure of the environment, it is very inefficient: the number of states is quadratic in the maximum length of the edges.

As a compromise between fidelity and efficiency, we have chosen to model edges by collapsing the separate chains into one. Figure 4c shows one possible way of collapsing the chains (here, of an edge with minimum length three and maximum length five), in a way that we call the "come from" semantics. 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 the chain is uncertain (e.g., state B is 1 meter from state A, but is between 2 and 4 meters from state C). If the distance from a state to the start state is less than the minimum length of the edge, or if the state is the last of the chain, then the forward transition probability out of that state is deterministic; Otherwise, there is some probability that the forward action will transition to the next state in the chain and some probability that it will transition directly to the state corresponding to the next topological node. Note that if the robot turns around at some state s within the corridor and heads back towards the start node, the model will correctly predict that the robot will reach the start when it has traveled the same distance it took to reach state s in the first place. That is, even though the exact length of the edge is uncertain, the robot knows exactly how long it has traveled in coming from a given topological node.

A similar way of representing edges is to use "go to" semantics, in which the location of a state is specified relative to the topological node toward which the robot goes, but the distance from the start node is uncertain. While both the "come from" and "go to" models are equivalent for the purpose of projecting probabilities forward

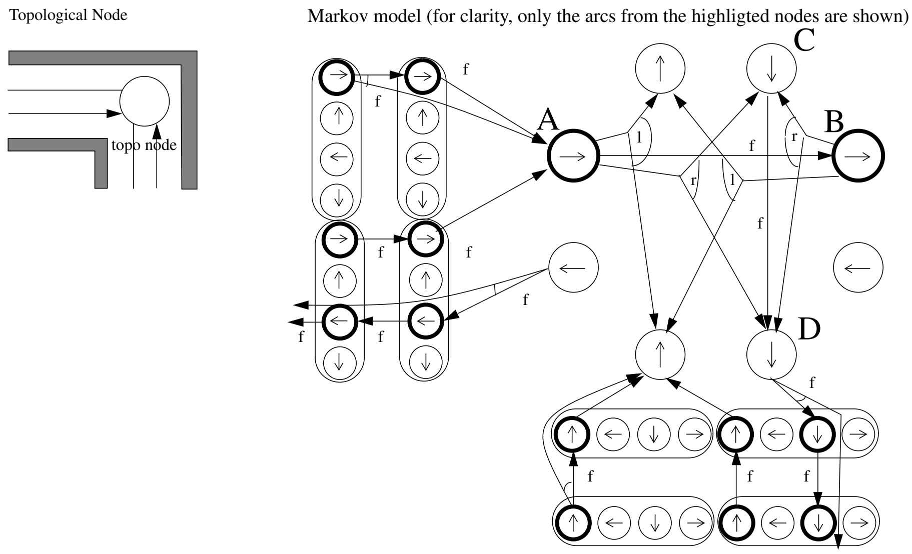

Figure 5: Representation of Topological Junctions

and can easily be compiled from the probability distribution over edge lengths, they have different advantages for the purposes of navigation. For example, using the "go to" semantics makes it easier to integrate virtual sensors (such as those based on vision) that detect features well ahead of the robot. On the other hand, the "come from" semantics can accurately model the ability to turn around within a corridor and return to the last topological node. In addition, the "come from" semantics seems more natural: it models the intuition that one is typically more uncertain about one's destination than in where one just came from. For these reasons, we use the "come from" semantics in our model.

We also need to model topological nodes. Our first representation consisting of a single group of four Markov states (as in Figure 3) proved somewhat inadequate, since the spatial resolution of a Markov state is one meter, but corridors are actually two meters across. While a straightforward fix would be to represent junctions using four (two by two) groups of four states each, we achieve nearly the same result with four groups of only two states each, which both saves space and makes the model much simpler (Figure 5). The basic idea is that turns within a junction are not modeled deterministically, but instead have 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 5 from the West, the robot would first encounter state A, then state B. If it then turned right, it would be facing South, and would transition to either states C or D with equal probability. While there are some minor inaccuracies with this representation (for instance, if the robot turns right in B and then left again, it is now with equal probability in A or B), in general it captures well how the robot actually traverses corridor intersections. In particular, it corresponds to the fact that it is very difficult to pin down the robot's location exactly while it is turning in the middle of an intersection.

We have made several extensions to the basic model described above to make it more generally useful for office navigation. First, to model dead-reckoning uncertainty we include a self-transition, that is, a transition with some probability from each state into itself, for all the forward actions. Doorways are modeled as exact-length (Figure 4a), two meter long edges. In addition, similarly to [12], we associate a probability p with the edge that indicates the likelihood that the door is open. Then, the observation probabilities associated with seeing a doorway are:

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

5. Position Estimation

As the robot moves, it receives information that is used to update the probability distribution over the Markov states. One source of information is dead-reckoning, which provides the current heading of the robot and the distance traveled (its "virtual odometer"). The navigation system is notified whenever the heading is changed, and it periodically receives distance reports from the virtual odometer. The action update rule presented in Section 3 is used to incorporate this information into the Markov model.

In general, the action of moving forward tends to increase positional uncertainty, due to non-deterministic transitions in the model. In some cases, however, knowing that the robot moved forward can decrease uncertainty. This occurs where there is some probability that the robot is a in state in which the forward action is not feasible (forward $\notin A(s)$). Such states are prevalent — for instance, all states within a corridor whose orientation is perpendicular to the axis of the corridor (see Figure 4). 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.

The other way to reduce positional uncertainty is through sensor reports. A sensor report consists of the feature detected, as well as the odometer reading and heading of the robot at the time of detection. The navigation system uses the action update rule, as described above, to move the model forward to the given distance, and then uses the observation update rules presented in Section 3 to incorporate the sensor information.

There are two practical problems that must be dealt with. First, sensor reports may be out of date: either the odometer reading associated with the report has been passed, or the heading reported is not the same as the current heading. This problem arises because, since the evidence grid integrates physical sensor information over time, the sensor reports are sometimes delayed up to the point where the the Markov model already incorporates actions that were executed after the sensor readings. Given only the current probability distribution and the delayed sensor report, it is impossible to update the probability distribution correctly. Currently, we just ignore such reports, but that loses information. A better approach would be to periodically record snapshots of the probability distributions and then, when a delayed sensor report is received, to revert to an earlier snapshot and redo all calculations from there, this time taking the sensor report into account at the right time.

The other problem with handling sensor reports is that occasionally when more information is integrated into the evidence grid, a previous sensor report is found to be erroneous and a new one is issued instead. If no action update has occurred between the old and new reports (and if all conditional probabilities of the sensor are non-zero), one can easily undo the effects of the old report and then incorporate the new one, as per usual. Given the current probability distribution $p(s)$ for all $s \in S$, the old report o of sensor i, and the new report o', one updates as follows:

$$p_{posterior}(s) = K \times \frac{p_i(o'|s)}{p_i(o|s)} \times p(s)$$

where K is a normalizing factor. If intervening action updates have occurred or at least one conditional probability $p_i(o|s)$ is zero, then the problem can be solved in the same way as the one above.

6. Planning and Action Execution

For self localization, the position estimation mechanism can be combined with a simple wandering behavior, until positional uncertainty is sufficiently reduced. To get the robot to move to a given goal, however, a planner assigns a desired action with each Markov state and, after each model update, the navigation system chooses the action with the highest total probability mass (the "best-action" strategy).

Given the large number of states, it is infeasible to plan directly using the Markov model. Rather, we plan using the augmented topological map and then map that plan to the Markov states. For each node and edge in the map, the planner associates a preferred direction to travel, based on the expected total travel distance to the goal. Then, the "forward" action is assigned to each Markov state whose orientation corresponds to the preferred direction of its associated topological entity. The remaining states are assigned actions that will turn the robot to the desired heading. For example, if the preferred direction at a topological node is North, then the following actions are assigned to the corresponding Markov states: go forward in the Markov states with orientation North, turn left in the states with orientation East, and turn right in the Markov states with orientation West or South (although the last action assignment is arbitrary — turning left is just as reasonable). Finally, a stop action is assigned to the goal state and to nearby states (the latter decreases the time until the stop action has the highest probability mass of all actions when the robot has reached the goal).

While we model the actions as discrete turns and moves, in actuality, the robot does not stop to turn, but continually moves forward while turning. In addition, turns are cumulative, so that two successive right turn actions, for instance, results in a smooth 180 degree turn. Since the robot continues to move forward in the direction

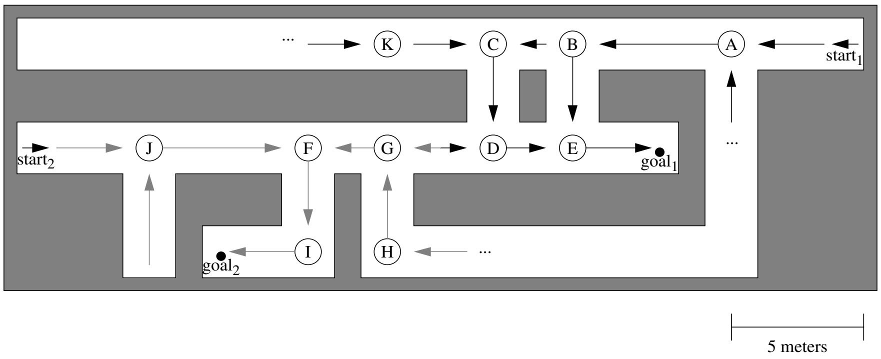

Figure 6: An Office Corridor Environment

of its current heading at all times (modulo detours to avoid obstacles), the forward action is never actually sent to the robot (unless it is stopped).

One variation on the "best-action" selection strategy is to choose an action only if its probability mass is above some threshold, otherwise the robot continues along its current heading (the "best-action-above-threshold" strategy). An advantage of this strategy is that it reduces the chance of making a wrong move due to spurious false positive sensor reports (e.g., seeing an opening where none exists), since several confirming reports will be needed to push the action over threshold. Also, if no action is clearly best, it may indicate that the robot is lost and needs to self-localize. On the other hand, waiting until a threshold is reached may delay the robot from taking the best action. In our simulation experiments, described below, the "best-action-above-threshold" strategy, with an action threshold of 80%, performed worse than the "best-action" strategy. However, we believe that using an action threshold may be more useful on our real robot, since the simulator tends not to hallucinate openings, while the real robot does this with some regularity [14].

7. Experiments

In this section, we demonstrate the performance of our navigation technique in a prototypical office corridor environment. To perform the experiments, we used a highly realistic simulation environment of Xavier, our mobile indoor robot [13]. Xavier is built on an RWI B24 base, includes bump sensors, sonars, a laser range sensor, and a color camera on a pan-tilt head. Control, perception and planning are carried out on two on-board, multi-processing 486-based machines. Xavier's nominal navigation speed is currently about 30 centimeters per second. Xavier is currently undergoing major hardware repairs which prevented us from performing experiments with the robot itself. However, we will do experiments similar to the ones described below with the real robot within the next two months and report the results in the final version of this paper. We also intend to compare its performance against the navigation system described in [14].

Two experiments were performed in the office corridor environment shown in Figure 6. The corresponding 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, and used a resolution of one meter to discretize the map. The resulting Markov model is relatively small: it has 1184 Markov states (the Markov model for half of one floor of our building has nearly 3000 states).

The initial positional uncertainty for both experiments is minimal: the initial probability for Xavier's actual location is about 90 percent. The remaining probability mass is distributed in the vicinity of the actual location.

In the first experiment, Xavier had to navigate from $start_1$ to $goal_1$. Our planner assigns the actions that are shown with solid arrows in the figure. Note that there is a choice of whether the preferred direction of travel between B and C should be East or West. Our planner decides to have the robot travel West (towards C) because this way it 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-action-above-threshold strategies, all of which were completed successfully. The results are shown in the following table:

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

The robot has to travel a rather long distance from the start before it has to make its first turn. Since this distance is uncertain and corridor openings are occasionally missed (such as the one at B) the robot occasionally overshoots B, and then becomes uncertain whether it is really at C or B. However, since the same action is assigned to both nodes, this ambiguity does not need to be resolved: Xavier turns left in both cases and then goes straight. The same thing happens when it gets to D or E. Now, however, the robot corrects its belief about its position when it then turns left, travels forward, and suddenly notices a corridor opening to its left. At this point in time, Xavier becomes fairly certain that it is at E. A purely expectation-driven landmark-based navigation technique can easily get confused in this situation. Although it could successfully follow the plan so far, suddenly it sees a left opening that should not be there and is confused as to whether this sensor report is wrong or whether its belief about its position is wrong. In the latter case, it cannot quantify its belief about were it might be instead.

In our second experiment, Xavier had to navigate from $start_2$ to $goal_2$. Our planner assigns the actions that are shown with dashed arrows in the figure. Again, we ran 15 trials for both action selection strategies.

For reasons that are similar to the ones in the first experiment, Xavier can confuse G with F. If it is at G but thinks it is 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 is 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 recover from misjudgements based on wrong sensor reports when it gets new clues as to where it is – even if it corrects its belief about its position very late and has to navigate back a long way. It is important to realize that this behavior is not triggered by any explicit exception mechanism, but results automatically from the way the plan execution and monitoring operates.

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

We emphasized earlier that an empirical validation of algorithms based on Markov models is important, because the world is not Markovian. Our experiments to date confirm our belief that our navigation technique operates robustly in office corridor environments, even in the presence of unreliable sensors and uncertainty in the distances.

8. Future Work and Conclusions

This paper has presented an approach to autonomous office navigation that uses partially observable Markov models to track the robot's position and to control its heading towards a goal. Advantages of this approach include the ability to account for uncertainty in the robot's initial position, actuator uncertainty, and sensor noise. Also, by integrating topological and metric information, the approach easily deals with uncertainty arising from incomplete descriptions of the environment.

We are pursuing several directions to extend this work. Obviously, we will do experiments on Xavier, when it becomes available. Foyers will be added to the model by using a 2D tessellation of the space. More fundamentally, we are exploring methods for passively refining the metric and topological map information contained in the models. We are currently applying conventional EM approaches for learning Markov models and, in parallel, developing our own, improved learning approaches that are more resistant to violations of the Markov assumption. We are also exploring probabilistic planning algorithms, such as approximations of POMDP algorithms and other, more efficient, algorithms that will take observation probabilities into account when choosing paths. For example, given two paths of equal length, the robot should prefer the one that minimizes the chance that it will miss a critical turn. Finally, we intend to add new sources of sensor information, primarily vision-based feature detectors.

Our approach enables a robot to utilize all its sensor information, both positional and feature-based, in order to robustly track its location. We have demonstrated that this approach leads to reliable goal-directed navigation, 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.

References

[1] Ronald C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proc. International Conference on Robotics and Automation*, Raleigh, NC, March 1987. IEEE.

[2] 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.

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

[4] H. Hu and J.M. Brady. A Bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots*, 1995. (submitted).

[5] 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.

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

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

[8] 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.

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

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

[11] 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.

[12] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish: An office-navigating robot. *AI Magazine*, 1995. (in press).

[13] J. O'Sullivan and K.Z. Haigh. Xavier manual. Technical Report Learning Lab Internal Document – contact josullvn@cs.cmu.edu, School of Computer Science, Carnegie Mellon University, 1994.

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

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

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