Learning Discrete Bayesian Models for Autonomous Agent Navigation

Daniel Nikovski and Illah Nourbakhsh

The Robotics Institute, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
email: {nikovski,nourbakhsh}@ri.cmu.edu

Abstract

Partially observable Markov decision processes (POMDPs) are a convenient representation for reasoning and planning in mobile robot applications. We investigate two algorithms for learning POMDPs from series of observation/action pairs by comparing their performance in fourteen synthetic worlds in conjunction with four planning algorithms. Experimental results suggest that the traditional Baum-Welch algorithm learns better the structure of worlds specifically designed to impede the agent, while a best-first model merging algorithm originally due to Stolcke and Omohundro [SO93] performs better in more benign worlds, including such that model typical real-world robot fetching tasks.

1 Introduction

Autonomous agents such as mobile robots often have to operate in non-deterministic environments with unobservable state and unknown dynamics. One typical example is the situation when a mobile delivery robot has to be deployed at a new site and is expected to navigate reliably between several predetermined goal locations. The only instruction it is given is a tour of the site, performed by joysticking it between all goal locations and allowing it to accumulate sensory-motor traces. When the robot is at a particular goal location, it is given an indication of that, including the goal number. After the tour is complete, the robot has to learn the topology of the site, plan routes between goal locations, and execute plans successfully. State in this environment is not fully observable because some of the locations are not distinguishable from each other, and furthermore, the robot is not certain about its starting position, the consequences of its actions are not deterministic, and its sensing is not perfect.

Classical planners such as STRIPS and UCPOP usually perform inadequately in such environments, because they have been designed to operate on deterministic representations of the world, available in advance in the form of a map or a domain theory, and require fully observable state. It has proved to be very difficult to extend classical planners to handle uncertainty and unobservability, while simultaneously learning the dynamics of the environment, and a major paradigm shift has been necessary.

A framework that can handle stochastic, partially observable worlds with unknown dynamics, is that of partially observable Markov decision processes (POMDP). While there has been considerable work on reasoning and planning with POMDPs [KLC96, Nou98, KS98], there has been relatively little progress on learning their structure from data with the explicit purpose of reasoning and planning with them thereafter. This is exactly the focus of this paper.

We apply two algorithms to the problem of learning POMDPs from data – the traditional Baum-Welch algorithm and a best-first model merging method adapted from the one proposed originally by Omohundro and Stolcke for speech recognition [SO93]. The performance of the three algorithms is compared in several planning domains using four algorithms for planning in POMDPs. The effect of the degree of aliasing is explored as well.

Section 2 describes the POMDP formalism and several popular algorithms for state estimation and planning with POMDPs. Section 3 discusses algorithms for learning in POMDPs and the problem of goal determination in learned POMDPs. Section 4 describes the fourteen worlds used in the experiments, as well as other experimental conditions. Section 5 reports the experimental results and section 6 concludes.

2 Reasoning and Planning with POMDPs

The POMDP model has been developed in the field of operations research as a formalism for inference and decision making in probabilistic systems with hidden state. A POMDP is described by the tuple (S, π, A, T, O, E, R), where S is a set of states, π is an initial probability distribution over these states, A is a set of actions, and T is a transition function that maps $S \times A$ into discrete probability distributions over S. The states in S are not observable – instead, observations from the set O can be perceived only. The function E maps $S \times A$ into discrete probability distributions over O (in some POMDPs the observations depend on state only). The function R describes the immediate reward received when the POMDP is in each of the states in S.

The structure, transition, emission, and reward functions are based on the control objectives of the agent that is controlling the POMDP. The scenario we are interested in is mobile robot navigation, where each state is a location in a world and each observation is a discrete percept that can be produced by the perceptual apparatus of a mobile robot. If the objective of the robot is to reach a particular goal state and stay there, the reward for that state can be, for example, one, while the reward for being in all other states can be zero. Then, if the agent maximizes the reward it receives, it will effectively be achieving its goal.

A POMDP can be controlled by executing one of the available actions at each time step. As a result, the POMDP transfers to a new unobservable state and emits a new observation which can be perceived by the agent that is controlling the POMDP. The agent has to infer the state of the POMDP on the basis of the sequence of observations and the knowledge it has about the transition and emission probabilities of the POMDP. Since the agent has to reason under uncertainty, it cannot be completely sure about the exact state the POMDP is in; instead, it has to maintain a *belief state* $Bel(S)$ represented as a probability distribution over all states in S.

2.1 Belief updating

At all times, the agent maintains a belief distribution over all possible states it can be in, and updates it according to the actions it takes and the percepts it observes. This process of belief updating follows a two-staged prediction-estimation cycle [KS98]. During prediction, the agent estimates how the belief state at time t will change if a particular action a_{t-1} is performed at time $t-1$, based on the belief state before the action was performed and general knowledge about how actions affect states:

$$\hat{Bel}(s_t) = \sum_{s_{t-1}} P(s_t | s_{t-1}, a_{t-1}) Bel(s_{t-1}),$$

where $\hat{Bel}(s_t)$ is the estimated belief state at time t for all states s in S, $Bel(s_{t-1})$ is the belief state at time $t-1$, $P(s_t | s_{t-1}, a_{t-1})$ are transition probabilities of the POMDP, and the sum runs over all states in S.

During the estimation phase, the newly observed percept o_t is used to update the estimated belief vector:

$$Bel(s_t) = \alpha P(o_t | a_t) \hat{Bel}(s_t),$$

where $P(o_t | s_t)$ are the emission probabilities of the POMDP and α is a normalization coefficient. The most likely state at each step is the one whose corresponding element of the belief vector is largest.

2.2 Planning with POMDPs

Controlling a POMDP amounts to choosing the correct actions that will maximize the expected cumulative reward received by the agent over the course of its operation. When this course is infinite in duration, the sum itself will be infinite – one way to avoid this is to apply an exponential discounting factor $\gamma < 1$ to each reward. Thus, the goal of the controller is to maximize the quantity $< \sum_{t=0}^{\infty} \gamma^t R_t >$, where R_t is the reward received at the t-th step of operation, and $< \cdot >$ denotes expectation. Many algorithms for choosing proper actions in POMDPs have been studied recently [CKK96], some of which are reviewed below.

2.2.1 Optimal control in POMDPs

One approach to controlling a POMDP that leads to optimal control is to transform the POMDP into a fully observable MDP, whose states are the belief states of the original POMDP. Since the space of all belief states is continuous and most methods for solving MDPs operate on discrete state, the belief state is usually discretized. However, applying this method directly is extremely demanding computationally and inefficient. In general, the problem of finding optimal policies for POMDPs has been proven to be PSPACE hard [KLC96] and can only be used for trivial POMDPS with less than ten states.

More efficient algorithms exist as well, starting with the work of Sondik, subsequently improved by Kaelbling and Littman [KLC96]. Even though these algorithms offer dramatic improvements in computational efficiency, they are still applicable only to

POMDPs with several tens of states. Because of the computational difficulty of finding optimal solutions to POMDPs, various heuristic suboptimal strategies have been explored, among which assumptive planning and MDP-based approaches.

2.2.2 Assumptive planning

This heuristic strategy, proposed by Nourbakhsh [Nou98] performs full belief updating of its belief state based on the transition and emission probabilities, as described in the previous sections, but makes several simplifying assumptions when choosing an action. First, it assumes that it is in the most likely state with complete certainty, thus ignoring the possibility of being in other states. Next, it constructs a deterministic FSA from the transition and emission probabilities of the original POMDP and uses a general search algorithm such as iterative deepening to find a path in the FSA. Furthermore, the planner produces a list of percepts that ought to be seen if the plan is executed and the FSA is a true representation of the world. However, in most cases the percepts seen by the agent will differ from the expected ones, which is an indication that the plan is no longer valid and the agent is lost. If this is the case, the planner is called again to find a new plan based on the latest estimate of the most likely state.

2.2.3 MDP-based strategies

Another set of heuristic strategies solves first the underlying MDP and then makes use of that solution in conjunction with the estimated belief state [CKK96, KS98]. This corresponds to the usual approach taken in control theory, where the problems of state estimation and control are solved separately – a closed-loop control law is designed under the assumption that the state is completely observable and a separate observer is designed to estimate the current state.

The solution of the underlying MDP is an optimal policy that maps each state into the action that maximizes the expected future cumulative reward. In order to find that action, an auxiliary function $Q(s, a)$ is computed, whose meaning is the expected cumulative reward if action a is performed in state s and an optimal policy is followed thereafter. If the Q-function is known for a state, the optimal action a^* for that state is the one with highest Q-value: $a^*(s) = argmax_a Q(s, a)$. The Q-function can be found by means of Q-learning, an iterative Monte-Carlo reinforcement learning method [SB98]:

$$\hat{Q}^{t+1}(s, a) := (1-\eta)\hat{Q}^t(s, a) + \eta[R(s) + \gamma \max_b \hat{Q}^t(s', b)],$$

where $\hat{Q}^t(s, a)$ is an estimate of the Q-function at iteration t, η is a learning rate coefficient, which should decrease quadratically with time, and the state s' is sampled according to the transition probabilities for state s and action a. Once the MDP is solved, the Q-values can be used in several ways to approximate the optimal policy for the POMDP case [CKK96, KS98]:

Most likely state method: The most likely state is found, as described in the previous subsection, and the optimal action for that state is chosen. This method is suboptimal if there is still substantial probability that the system is not in the most likely state and the optimal action for the ignored states is different.

Voting method: This method chooses the action with highest probability mass in the belief vector. For each action, the belief probabilities for the states where this action is optimal are added up, and the action with the highest sum is chosen.

Q_{MDP} method: Similar to the voting method, but instead of adding up state beliefs only to the sum of the winning action, they are added to the sums of all actions, weighted proportionally to the actual Q-values. This avoids choosing the wrong action in cases when one action wins by a small margin in several states, but loses badly in another state. The Q_{MDP} method would be optimal if all uncertainty in the world was to suddenly disappear at the next time step.

All of the above methods based on the solution of the underlying MDP cannot plan to perform an action solely to gain more information. Various strategies have been researched that aim to reduce the degree of uncertainty as measured by the entropy of the belief vector [CKK96]. Another approach, the SPOVA-RL algorithm due to Parr and Russell, uses a special-purpose function approximator to estimate the expected reward of each state following the optimal policy for the POMDP [PR95]. It has been reported to solve successfully POMDPs with hundreds of states and actions.

3 Learning POMDPs

Learning a POMDP from observations instead of having one available in advance furthermore complicates the problem of determining optimal policies. Several fundamental questions arise, among which are the proper criterion to be optimized during learning, the correspondence between world and model states, and the related problem of goal determination.

There have been debates about the performance measure that should be maximized during learning – one point of view is that the available data should be used to learn all possible perceptual distinctions in the real world [Chr92], while the opposite view would have it that only those distinctions that are utile in actual planning and control should be learned [McC92]. Those algorithms that learn only utile distinctions use either decision trees or instance-based learning to form state representations. Since there are no known algorithms that learn POMDPs so that only utile distinctions are distinguished, we will consider only algorithms that learn all possible distinctions, that is, learn to predict training data as well as possible, without regard of how the learned POMDP will be used. Two such algorithms are described in the following subsections.

The correspondence between world and model states is another major issue in learning POMDPs from data. In general, the agent does not know how many states existed in the original process that generated the data and has to make a decision about the final number of states in the model. Furthermore, the original state space might have been continuous, while states in POMDPs are usually discrete. In such a case, a single learned state would correspond to many original states. Deciding how to establish a correspondence between continuous real and learned discrete states would require quantization of the continuous space without ever observing it directly, which is a difficult problem.

Another difficulty is that of determining the goal criteria for planning with the learned POMDP. Both assumptive planning and the MDP-based strategies need a goal state – either to terminate the iterative deepening search for assumptive planning, or to construct a proper reward function for the MDP-based algorithms. However, when a POMDP is learned, it is not clear which of its states correspond to the true goal states in the real world; in general, there won't be one-to-one correspondence between learned and true states at all. This circumstance changes significantly the goal criterion used for planning.

One possibility is to transfer the goal from the state domain to the perceptual domain, that is, instead of trying to reach a goal location, the agent tries to observe the percept that corresponded to that location in the training sensory-motor trace. For assumptive planning, the states of the FSA that is an idealized representation of the learned POMDP can be labeled with the most likely percept to be observed in that state. A solution exists for the MDP-based planners as well. Instead of assigning reward one to the goal state and zero to all other states, the reward for each state can be equal to the probability that the goal percept will be observed in this state. Thus, a policy that maximizes reward will in effect maximize the probability that the goal percept is seen; if the system is at the goal location each time the goal percept is seen, this policy will also maximize the probability of reaching the goal state.

Learning in POMDPs can be performed by means of general algorithms for learning probabilistic networks from data, such as those proposed in [HGC95, RBK94]. The goal of learning is to find values for the entries in the transition and emission conditional probability tables (CPT) of a POMDP, which maximize the log-likelihood $\ln P(\mathbf{D}|\mathbf{w})$ that the training data set $\mathbf{D}$ was generated by the POMDP with parameters $\mathbf{w}$. Each case D_l from the data set $\mathbf{D}$ consists of assignments for the observable nodes in O. Two such algorithms will be considered: the first one is the Baum-Welch learning rule, based on the expectation-maximization (EM) algorithm, and the second one is based on best-first model merging.

3.1 Baum-Welch

Koenig and Simmons [KS98] adapted the Baum-Welch algorithm for learning HMMs to the problem of improving the entries of the CPTs of POMDPs from data. The latter is a variant of the EM algorithm popular in statistics. The learning rule of the Baum-Welch algorithm is:

$$\hat{w}_{ijk}^{T+1} = \frac{\sum_l \xi_{i,l}^T(j,k)}{\sum_l \gamma_{i,l}^T(j)},$$

where $\hat{w}_{ijk}^{T+1}$ is the estimate of the transition probability $P[S_i(t+1) = s_{ik}|\mathbf{U}_i(t) = \mathbf{u}_{ij}]$ that node S_i will be in its j-th state s_{ij} given that its parents $\mathbf{U}_i$ are in their k-th configuration $\mathbf{u}_k$, after iteration $T+1$, and

$$\xi_{i,l}^T(j,k) = P[S_i(t+1) = s_{ik}, \mathbf{U}_i(t) = \mathbf{u}_{ij}|D_l, \hat{\mathbf{w}}^k]$$

$$\gamma_{i,l}^T(j) = P[\mathbf{U}_i(t) = \mathbf{u}_{ij}|D_l, \hat{\mathbf{w}}^k].$$

3.2 Best-first model merging

A completely different method for learning HMMs has been suggested by Stolcke and Omohundro, originally for the purposes of speech recognition [SO93]. Their approach consists of building an initial HMM, which has a separate state for each time step in the observation sequence. The transition and emission tables of this HMM are trivial – state t transitions to state $t+1$ with probability one, and emits observation $O(t)$ with probability one. Clearly, this HMM explains the observation sequence perfectly, but is not the most efficient model that can be learned from the data. In order to compact the model, pairs of states in the original HMM can be merged, so that the model retains its predictive abilities with respect to the observation data as much as possible. This can be achieved by considering several candidate mergers and choosing the one that decreases the likelihood of the model the least. After a merger is chosen, a new set of mergers is considered, and this process continues until the likelihood of the model with respect to the data becomes unacceptably low, or a predetermined number of states is reached.

This algorithm can be adapted to learning POMDPs, with several caveats. The transition tables of the initial POMDP are incomplete – for each state s, they contain entries only for the action that was actually taken at that time step. What happens if other actions are performed is unknown. Subsequent merges, however, are likely to produce full transition tables.

The original algorithm of Stolcke and Omohundro uses a Bayesian criterion for performing mergers of states, which can introduce various learning biases by specifying different prior distributions over POMDP structure and parameter values. If, however, all parameter values are equally likely, the Bayesian criterion is equivalent to choosing the mergers that decrease likelihood the least.

One particular advantage of this algorithm over the previous one based on iterative optimization, is the ability to use various heuristics in the merging process. For example, it is reasonable to merge states labeled with the same goal percept only among themselves, and not with states labeled with different percepts. In this way, one-to-one correspondence can be established between a learned state and a real goal state, which could facilitate planning.

4 Experimental environment

The previous sections described four planners and two learning methods, each combination of which will have relative strengths and weaknesses in different worlds. The goal of the experiments reported below is to investigate these and find whether there is a best combination that can be expected to perform well in a wide class of worlds.

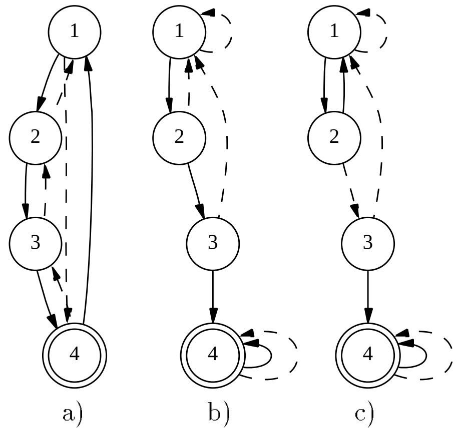

Figure 1: Test worlds: a)Easy4; b)Hard4; c) Harder4.

There are a number of things that make one world more difficult to learn and navigate than another. One of them is the probability of reaching the goal state by random choice of actions – the lower that probability, the harder the world. For example, the world in Fig. 1a is much more benign than the one in Fig. 1b because the length of a random walk to the goal is much longer in the latter than in the former. Another dimension of difficulty is the encoded length of the policy that is required for successful control. For example, the worlds in Figs. 1b and 1c have the same topology, but while performing action 1 all the time guarantees success in the former, the latter requires a much more detailed policy that depends strongly on the state the POMDP is in. A third dimension is the size of the world, which directly influences the difficulty of reasoning, planning, and learning. A related variable condition is the length of the action-observation sequence that the agent has available – the bigger the world, the longer a sequence will be required to capture its dynamics. Finally, the degree of perceptual aliasing in the world would substantially affect the agent's ability to learn a description of it and use that description in planning.

In order to test the performance of the planners and learning algorithms and the influence of the variable conditions described above, a number of experiments were performed, whose results are reported in the next section.

The four planners, described in the previous section, were tested on a total of fourteen worlds. There were only eight distinct transition diagrams, six of which for the worlds Easy, Hard, and Harder, with either four or eight states. Both aliased and non-aliased versions were tested. Two actions existed in each of these twelve worlds. The difference between these worlds is the probability that a random choice of actions will achieve the goal. The Easy world is very benign in that random actions are very likely to succeed. The Hard and Harder worlds are very punishing, because at each step a wrong action takes the agent back to the state farthest from the goal. The difference between Hard and Harder is that in Hard the correct policy is to always take action one, no matter what percepts are observed, while in Harder the correct action depends strongly on the state the agent is in, which can be identified only by observing percepts. The agent can observe the state number, except if aliasing is introduced, in which case states one and two emit the same percept. All transitions are deterministic.

The last two worlds, Gold 6 and Gold 12, represent worlds where a robot has to find a piece of gold in one room and carry it to a specific goal room. Gold 6 has two rooms and hence six states (the gold can be in either room or carried by the robot); Gold 12 has three rooms and twelve states. Four actions are possible – movement in two directions, clockwise and counterclockwise, as well as grasping and dropping the piece of gold. The robot can perceive the room number and whether the piece of gold is either in the room, not in the room, or in its hand. Hence, a total of six percepts can be observed in Gold 6 and nine in Gold 12, which makes the former world fully observable, while the latter is aliased. A human placed in the Gold 12 world would first plan to find the gold, and then carry it to the goal. However, the planners considered here cannot plan to perform actions just in order to change their belief state, which makes this world hard for them.

Each of the worlds has one goal state, and the results are averaged for all possible starting states. The cumulative discounted reward has a discount factor $\gamma = 0.9$. If a planner cannot achieve the goal within 250 steps, the run is terminated and zero award is given.

The MDP-based planners solved the underlying MDP of the learned POMDP by Q-learning, sweeping each state in turn and sampling successive states according to the transition probabilities of the MDP. Learning rate $\eta = 0.1$ and discounting factor $\gamma = 0.9$ were used, for a total of 1000 sweeps. Each experiment used a sequence of 100 observations, except where indicated differently.

5 Results

Closely following the methodology of [CKK96], Table 1 shows the discounted cumulative reward for achieving the true goal state averaged over each initial state for fourteen worlds, two learning methods, and four planners. Average results across planners are shown as well. The achieved reward for random actions is listed as a baseline for comparison. Each number is the mean over ten trials, and the variances from these trials are used to test for statistically significant differences between performances. All combinations of learning and planning methods were tested against random choice – those results, which are significantly better than random walk at the 5% error level, are shown in bold. It can be claimed that in these cases the learner/planner does indeed learn an appropriate model of the world and uses the learned model to choose actions deliberately.

Paired t-tests for significant differences between the two learning methods for the same world and planner were performed as well. Whenever one of the learning methods significantly dominates the other one for a given world and planner at the 5% error level, the achieved reward is shown in a frame. It can be seen that BFMM is generally better for simpler worlds, while Baum-Welch dominates the harder ones. However, BFMM does much better in the aliased Gold 12 world, which is closest of all worlds to a real robot application. The performance of Baum-Welch is not affected from the length of the observation sequence in the world Harder 8 NA, while BFMM greatly improves its performance when 300 instead of 100 observations are used.

6 Conclusion

The reported results confirm the expectations that both Baum-Welch and BFMM are susceptible to local maxima and hardly ever recover fully the original POMDP that generated the observations. However, in most cases they recover enough of the POMDP to be able to perform better than random choice of actions would. Furthermore, the complexity of the world does not seem to affect this tendency. Overall, BFMM performs better on simpler worlds, while Baum-Welch seems to handle better more complex ones. BFMM clearly dominates Baum-Welch on the aliased pick-and-drop task, which suggests that real world navigation tasks are usually benign and BFMM might be the better choice for them.

World	Baum-Welch					Best-first model merging					Rnd
	AP	MLS	Vote	Q_{mdp}	Avg	AP	MLS	Vote	Q_{mdp}	Avg	
Easy 4 NA 100	0.72	**0.61**	0.75	0.75	0.71	$\boxed{\mathbf{0.87}}$	$\boxed{0.82}$	0.82	0.82	0.83	0.79
Easy 4 AL 100	0.82	0.72	0.79	0.79	0.78	0.85	$\boxed{\mathbf{0.89}}$	$\boxed{\mathbf{0.89}}$	$\boxed{\mathbf{0.90}}$	0.88	0.83
Easy 8 NA 100	0.49	0.63	0.64	0.62	0.60	$\boxed{\mathbf{0.69}}$	**0.74**	**0.74**	**0.74**	0.73	0.58
Easy 8 AL 100	**0.73**	**0.81**	**0.81**	**0.81**	0.79	**0.70**	**0.73**	**0.75**	**0.75**	0.73	0.51
Hard 4 NA 100	0.67	**0.74**	**0.74**	**0.74**	0.72	**0.75**	0.58	0.58	0.58	0.62	0.57
Hard 4 AL 100	0.67	0.64	0.68	0.68	0.66	0.54	0.72	0.72	0.72	0.68	0.60
Hard 8 NA 100	0.38	$\boxed{\mathbf{0.71}}$	$\boxed{\mathbf{0.71}}$	$\boxed{\mathbf{0.71}}$	0.63	0.33	**0.48**	**0.48**	**0.48**	0.44	0.26
Hard 8 AL 100	0.28	$\boxed{\mathbf{0.66}}$	$\boxed{\mathbf{0.66}}$	$\boxed{\mathbf{0.66}}$	0.57	0.35	0.37	0.37	0.37	0.36	0.24
Harder 4 NA 100	0.62	0.51	0.51	0.51	0.53	**0.47**	0.55	0.55	0.55	0.53	0.62
Harder 4 AL 100	$\boxed{0.61}$	0.50	0.50	0.50	0.53	**0.43**	**0.45**	**0.45**	**0.45**	0.44	0.58
Harder 8 NA 100	0.28	$\boxed{0.32}$	$\boxed{0.32}$	$\boxed{0.32}$	0.31	0.26	0.19	0.19	0.19	0.21	0.26
Harder 8 AL 100	**0.38**	$\boxed{\mathbf{0.50}}$	$\boxed{\mathbf{0.50}}$	$\boxed{\mathbf{0.50}}$	0.47	**0.36**	0.31	0.31	0.32	0.33	0.23
Harder 8 NA 300	**0.20**	0.31	0.31	0.32	0.28	$\boxed{0.31}$	**0.40**	**0.40**	**0.40**	0.37	0.26
Gold 6 NA 100	$\boxed{\mathbf{0.74}}$	**0.80**	**0.80**	**0.80**	0.78	**0.63**	**0.71**	**0.71**	**0.71**	0.69	0.30
Gold 12 AL 300	**0.30**	**0.30**	0.26	0.26	0.28	**0.35**	$\boxed{\mathbf{0.55}}$	$\boxed{\mathbf{0.55}}$	$\boxed{\mathbf{0.55}}$	0.50	0.22

Table 1: Cumulative rewards for fourteen test worlds, two learning methods, and four planners, $\gamma = 0.9$. All experiments used 100 observations, except Gold with 3 rooms, which used 300, and Harder 8, non-aliased, which used both 100 and 300 observations. NA denotes non-aliased, while AL aliased worlds.

References

[Chr92] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 183–188, San Jose, CA, July 1992. MIT Press.

[CKK96] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[HGC95] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[KLC96] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. Technical Report CS-96-08, Brown University, Providence, RI, 1996.

[KS98] Sven Koenig and Reid Simmons. Xavier: a robot navigation architecture based on partially observable Markov decision process models. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 91–122. MIT Press, Cambridge, 1998.

[McC92] R. A. McCallum. First results with utile distinction memory for reinforcement learning. Technical Report 446, Computer Science Department, University of Rochester, NY, December 1992.

[Nou98] Illah Nourbakhsh. Dervish: an office navigating robot. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 73–90. MIT Press, Cambridge, 1998.

[PR95] Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[RBK94] Stuart Russell, John Binder, and Daphne Koller. Adaptive probabilistic networks. Technical Report CSD-94-824, University of California, Berkeley, July 1994.

[SB98] R. S. Sutton and A. G. Barto. *Reinforcement learning*. The MIT Press, Cambridge, Massachusetts, 1998.

[SO93] Andreas Stolcke and Stephen Omohundro. Hidden Markov Model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18. Morgan Kaufmann, San Mateo, CA, 1993.