Finding Approximate POMDP solutions Through Belief Compression

Thesis Proposal

Nicholas Roy
Carnegie Mellon University

Abstract

Human beings take for granted their ability to move around and interact with each other in a wide range of environments. These same environments are tremendous sources of uncertainty for mobile robots. Not only does navigation introduce positional uncertainty, but systems that try to interact with human beings are faced with the tremendously noisy and ambiguous behaviours that humans exhibit. More recent planners have been developed that to handle the non-deterministic and unexpected outcomes of actions. For example, a number of probabilistic planners model the real world as a Markov process. However, most such planners assume that the true state of the world can always be determined accurately, an assumption that is . In an world without enough structure, tracking the current position of a robot is hard; the sensors cannot reliably track the environment as the robot moves.

The Partially Observable Markov Decision Process (POMDP) constitutes a particular planning methodology that attempts to model uncertainty in the way that a system acts and senses the world. Instead of making decisions based on the current perceived state of the world, the POMDP maintains a belief, or probability distribution, over the possible states of the world, and makes decisions based on the current belief. POMDPs are unfortunately useless for most real world problems due to the overwhelming computational complexity involved with planning in belief spaces. However, the uncertainty that many real world problems exhibit often has a regular structure that can be exploited. By making assumptions about the kinds of beliefs that a system may have, it may be possible to reduce the complexity of finding policies that approximate the optimal POMDP policy, without being subject to the pain of finding the exact optimal policy.

I propose a method for approximating optimal POMDP policies by using a compressed representation of the full belief state. This representation, the augmented MDP, essentially parameterises the belief state; in the example domains in this work, the belief space is parameterised as the most-likely state, and the entropy of the belief state. By discretising the parameters appropriately traditional MDP techniques for value iteration can be used to find a policy in the belief space.

As evidence of the utility of this approach, I will show preliminary results of using the augmented MDP in two different domains. The first domain is mobile robot navigation, where my approach generates motion policies that minimise localisation error. The second domain is speech dialogue management, and I demonstrate the use of the augmented MDP in generating policies that are robust to speech recognition error.

Introduction

Human beings take for granted the ability to move around and interact with each other. For example, people generally do not get lost in familiar environments. Unfamiliar environments can be a problem, so many places have signs that help newcomers. People can also understand one another easily. Although most conversations contain tremendous ambiguity, such as wildly varying speech and acoustic signals that all correspond to the same ideas, people manage to communicate quite readily.

Robots, and machines in general, do not yet have the same kinds of abilities as human beings. Robots do navigate, and some even navigate using vision, but not yet with the same flexibility and reliability as humans do. Similarly, human robot interaction is typically characterised by an unnatural rigidity.

Many of the problem that robots encounter in the real world are a consequence of the assumptions that are implicit in the planning and control algorithms that the robots use. Traditional, logical planning algorithms have typically assumed perfect models of both the world and the robot, not only at an abstract level but at every level of control. For example, early motion planners assumed that the robot could be modelled with deterministic actions; given a command to go forward one meter, the robot would do so exactly and would do so every time. However, the real world does not behave as expected, and in fact does not behave predictably. Wheels skid, gears slip, people appear in unexpected places and ask questions in ways they have never done before.

More recent planners have been developed that handle the non-deterministic and unexpected outcomes of actions. For example, some probabilistic planners model the real world as a Markov process, and have been used successfully in a number of applications. Other planners handle the problem differently, such as using purely reactive control at the lowest level. The probabilistic framework is appealing for a number of reasons; probabilistic models allow a system to reason about the expected effect of actions in ways that other methodologies cannot.

Nevertheless, existing planners still make assumptions that do not necessarily reflect real world conditions. When a system takes an action, it is usually prepared for a number of possible outcomes, but the system assumes that these outcomes can be detected accurately and completely. Detecting the outcome of an action may be require deliberate sensing, but rarely, if ever, is the final state of the world called into question. Most systems consider the state of the world to be accessible (Russell and Norvig, 1995).

However, the real world is rarely, if ever, accessible. Determining the true state of the world, or the state of the system with respect to the world, can be difficult and may not always be possible. For example, in an environment without enough environmental structure, tracking the current position of a robot is hard; the sensors cannot reliably track the environment as the robot moves. Imagine a robot moving across a vast desert. After a while, the robot becomes totally lost, because as the wheels skid and the gears slip, there is nothing that the robot can sense to indicate that it isn't moving in the expected way. The right approach is to avoid such situations as much as possible. The world consists largely of situations where the current state is easy to find, combined with some situations where the current state is hard to track[1].

The key point is that sensing is an imperfect operation; sensing the world must be done to identify the posterior state after acting, but the sensor does not always do a good job of identifying that state. A number of planning methodologies lend themselves easily to modelling explicitly sensing actions, especially cost-benefit analysis of expensive sensing operations, but very few model how informative the sensing act will be.

A simple example can show how useful it can be to model sensing fully. Imagine a robot attempting to cross a large expanse, perhaps a large field, or a foyer in a building. Most planners will plan to cross the expanse directly; this is the optimal path in terms of minimising distance or time. However, the robot may become lost, or even completely turned around if its odometry is very poor. The most reliable trajectory will be to skirt around the edges of the open space, keeping close to the environmental structure near the edges. While such a path is sub-optimal in terms of distance and time, it will offer the best guarantee of reaching the goal.

We can see another example in human-computer interaction. Human speech is often ambiguous, such as in *"Pick up that box."* If the robot models the state of the world as the intention of the user, it becomes impossible to identify the true state when faced with multiple boxes. A conventional planner may have no way of disambiguating the possible

[1] Problems where the current state is impossible to track everywhere are about as rare and pathological in the real world as problems where the current state can be found correctly every time.

hypotheses; a planner that models the effect of ambiguous sensing can act to disambiguate the user's intention by asking for confirmation of which box.

Such a planning methodology does in fact exist; the Partially Observable Markov Decision Process (POMDP) maintains probabilistic model of sensing and acting (Sondik, 1971). Whereas the simple Markov Decision Process (MDP) models the problem of unpredictability in how the world changes as a result of actions, the POMDP models the problem of not even being able to determine what the state is. Instead of making decisions based on the current perceived state of the world, the POMDP maintains a belief, or probability distribution, over the possible states of the world, and makes decisions based on the current belief. A POMDP policy will make exactly the right kind of compromise between conventional optimality considerations and the certainty of achieving the goal state.

POMDPs are unfortunately useless for most real world problems due to overwhelming computational complexity involved with planning in belief spaces. However, the uncertainty that many real world problems exhibit often has a regular structure that can be exploited. By making assumptions about the kinds of beliefs that a system may have, it may be possible to reduce the complexity of finding policies that approximate the optimal POMDP policy, without being subject to the computational cost of finding the exact optimal policy.

I propose a method for finding an approximation of the optimal POMDP policy by compressing the full belief state into statistics that represent the belief space compactly. This representation, the augmented MDP, essentially parameterises the belief state; by discretising the parameters appropriately traditional MDP techniques for value iteration can be used to find a policy over the belief space.

I will show formally how certain POMDPs may be converted into an augmented MDP. Preliminary results use a restrictive set of assumptions; part of this proposal is to develop a way of finding general representations for representing arbitrary POMDPs. I furthermore propose to examine the quality of the approximation, to determine how badly the approximate policy suffers, in exchange for computational tractability.

I will also demonstrate some results of applying the augmented MDP in two different domains. The first domain is mobile robot navigation; the new technique generates motion policies that minimise localisation error. The second domain is speech dialogue management, and the augmented MDP is used to generate policies that are robust to speech recognition error.

2. Probabilistic Decision Making

Real world processes are inherently non-deterministic. When the world changes, either by itself or because of some process acting on the world, the new state of the world is hard to predict. Markov Processes are a particular way of modelling non-deterministic changes in the world, under the assumption that the new state of the world can be predicted from the current state of the world; knowing the history that led to the current state does not improve the estimate of future states (Howard, 1960).

2.0.1 Markov Decision Processes Markov Decision Processes (MDPs) are a planning mechanism, that describe how to act in a Markovian world. A Markov Decision Process is given by the following:

- a set of states $\mathcal{S}\epsilon\{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_n\}$
- a set of actions $\mathcal{A}\epsilon\{a_1, a_2, \ldots, a_m\}$
- a set of transition probabilities $T(\mathbf{s}', a, \mathbf{s}) = p(\mathbf{s}'|\mathbf{s}, a)$
- a set of rewards $R : \mathcal{S} \times \mathcal{A} \mapsto \Re$
- a discount factor $\gamma\epsilon[0, 1]$
- an initial state $s_o\epsilon\mathcal{S}$

The transition probabilities form a structure over the set of states, connecting the states in a directed graph with arcs between states with non-zero transition probabilities. The discount factor represents the lifetime of the agent; in a sense, it represents the probability that the agent survives to take another step.

A policy for an MDP is a set of state-action pairs that describe, for each state, which action to take. The optimal policy is the policy that the agent should follow to maximise the expected reward over the lifetime of the agent and is known to exist in the discounted ($\gamma < 1$) case with bounded immediate reward (Howard, 1960).

The optimal policy is found by first computing a value for each state that corresponds to the expected reward for that state. The expected reward is computed with respect to the optimal policy – it is the reward the agent expects to receive over its lifetime if it follows the optimal policy. The value function is found using Bellman's equation (Bellman, 1957):

$$J(\mathbf{s}_i) = \max_a \left(R(\mathbf{s}_i, a) + \gamma \sum_{j=1}^{N} p(\mathbf{s}_j|\mathbf{s}_i, a) \cdot J(\mathbf{s}_j) \right) \tag{1}$$

This equation is defined recursively, and is found through iterating over all states repeatedly until the value function converges to some fixed point. The optimal policy is then to take the action that leads to the next state with highest expected reward:

$$\pi(\mathbf{s}_i) = \operatorname*{argmax}_a \left(R(\mathbf{s}_i, a) + \gamma \sum_{j=1}^{N} p(\mathbf{s}_j|\mathbf{s}_i, a) \cdot J(\mathbf{s}_j) \right) \tag{2}$$

It should be noted that the exact form of Bellman's equation may differ from problem to problem, for example the immediate reward $R(\mathbf{s}_i, a)$ may depend only on the state: $R : \mathcal{S} \mapsto \Re$.

Although MDPs capture the non-deterministic aspects of actions in the world, they do not address a different problem faced by real world systems, and that is the problem of observability. The real world is not only non-deterministic, but is usually only partially observable at best. It is rarely possible to know the true state of the world; the best one can do is to make observations of the state of the world that can be noisy and ambiguous. An observer can often make *some* conclusions about the state of the world, but not usually with 100% certainty.

Real world processes that suffer from partial observability (or even complete unobservability) are usually modelled by Hidden Markov Models (HMMs) (Rabiner, 1990). The underlying state transitions of the HMM obey the Markov property as stated above, but the current state at any point in time is not known, and can only be inferred from observations over time.

2.0.2 Partially Observable Markov Decision Processes Partially Observable Markov Decision Processes (POMDPs) are a framework for planning in partially observable worlds (Sondik, 1971; Cassandra et al., 1994). At no time is a system acting in the world guaranteed to know the state of the world, and the system may not even know its own state with respect to the world (e.g., its exact position may be only partially observable).

The POMDP consists of an underlying, unobservable Markov Decision Process, as given above, with the addition of observations and observation emission probabilities:

- a set of states $\mathcal{S}\epsilon\{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_n\}$
- a set of actions $\mathcal{A}\epsilon\{a_1, a_2, \ldots, a_l\}$
- a set of transition probabilities $T(\mathbf{s}', a, \mathbf{s}) = p(\mathbf{s}'|\mathbf{s}, a)$
- a set of observations $\mathcal{Z}\epsilon\{z_1, z_2, \ldots, z_m\}$
- a set of observation probabilities $O(z, \mathbf{s}, a) = p(z \,|\, \mathbf{s}, a)$
- an initial belief, $p_0(\mathbf{s} : \mathbf{s}\,\epsilon\mathcal{S})$
- a set of rewards $R : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \mapsto \Re$

Notice that instead of an initial state, the POMDP is initialised with an initial belief, that is, a probability distribution over the state space. If the system knows exactly which state it is in, this belief is a delta function.

The notion of beliefs is at the crux of how a POMDP operates. Whereas an MDP policy dictates an action for every state, the POMDP dictates an action for every possible belief. Since the belief space is continuous, the value function is defined over a continuous space, making dynamic programming substantially more computationally complex than its MDP counterpart.

The value function is also piece-wise linear and convex, a consequence of how the value function is constructed. The value function over the belief space is composed of the superposition of hyperplanes. Recall that the value function is the expected (discounted) reward over the lifetime of the agent as given by the horizon. Each hyperplane, or vector, represents this expected reward for a particular sequence of actions and observations. This sequence represents a policy and a possible set of observations that could be received while executing that policy. The exact equation of the vector is a function of the expected reward for following the policy given by that vector, and getting observations given by that vector. The vector can also be viewed as a linear interpolation of the immediate rewards at every state weighted by the probability of arriving at each state given the action-observation sequence.

2.1. Finding POMDP policies

Littman showed a number of complexity bounds for POMDPs (Littman, 1996); these bounds demonstrate the intractability of finding an exact solution for those POMDPs where a solution is known to exist. The general case of an infinite-horizon, stochastic POMDP is EXPTIME-hard for boolean rewards, and is not known to be decidable for general (but bounded) rewards. Stochastic POMDP problems with a finite horizon are PSPACE-hard (PSPACE-complete for boolean rewards), and a deterministic POMDP with a finite horizon is NP-complete.

A discussion of exactly why POMDPs are so complex is outside the scope of this paper. However, it is possible to get an intuition into the problem. For a given sequence of n actions there are n observations. If there are m possible observations, then there are m^n possible action-observation sequences for a given sequence of actions. Furthermore, if there are k possible action sequences, then there are $k^n m^n$ possible action-observation sequences. Each of these action-observation sequences is in fact a policy, and in the worst case may be the optimal policy for some (extremely small) segment of the belief space. Some policies are usually strictly dominated and can be discarded, but no such claims can be made in general. In fact, for an infinite horizon problem, there are potentially infinitely many such sequences that represent the optimal policy. It is worth noting that Monahan's enumeration algorithm (Monahan, 1982) operates in this manner, by enumerating such sequences of action-observation pairs, computing the value vectors and removing dominated vectors.

There are a number of techniques and heuristics for finding the approximations to the value function or policies that do mostly the right thing. Perhaps the most well-known of these is Witness algorithm (Kaelbling et al., 1998). Witness

is an iterative algorithm that is capable of finding an exact solution to the value function if allowed to run to convergence, although typically it is used as an approximate algorithm through early termination. Witness approximates the value function with vectors from the true value function, and then searches for points (witnesses) where the vector no longer dominates. The Smooth Partially Observable Value Approximation (SPOVA) algorithm (Parr and Russell, 1995) is an example of an approximation algorithm that also approximates the value function. SPOVA computes the Bellman residual for randomly sampled belief states, and adjusts the parameters of the fixed set of vectors until some termination condition is met. The Monte Carlo POMDP (Thrun, 1999) is another approximation algorithm that represents the policy using a sample-based representation. The MC-POMDP performs a number of one-step roll-outs on samples of the belief space, and computes the expected reward of taking an action from each sample. The value for an arbitrary belief is then found by linear averaging from the k-nearest neighbours.

2.2. Heuristics

There is a family of policies that operate in partially observable worlds, but sidestep the computational intractability of finding an optimal solution by using greedy heuristics. These heuristics can all be made to fail, but how and why they operate successfully can lend insight into alternative approaches.

The first and simplest heuristic is the "most likely state" heuristic (Nourbakhsh et al., 1995). During planning, only the underlying MDP is solved, using the standard MDP solution technique. During execution, a distribution over states is maintained, but at every step the agent uses the action dictated by the MDP at the most likely step. This heuristic has been used successfully in robot navigation; however, it depends on the fact that the most likely state in the belief space is indeed the true state. This heuristic ignores competing hypotheses represented in the belief, which can lead to sub-optimal policies when the most likely state is wrong.

The policy described by "most-likely state" heuristic for a particular belief $p(\mathbf{s})$ can be represented as:

$$\pi(p(\mathbf{s})) = \operatorname*{argmax}_a(Q(\operatorname*{argmax}_{\mathbf{s}}(p(\mathbf{s})), a)) \tag{3}$$

where $Q(s, a)$ is the expected reward one action later after taking action a, and is given by:

$$Q(\mathbf{s}, a) = \gamma \sum_{j=1}^{N} p(\mathbf{s}_j|\mathbf{s}, a) \cdot J(\mathbf{s}_j) \tag{4}$$

where $J(\mathbf{s}_j)$ is the value for the MDP at state $\mathbf{s}_j$ given by equation (1). Although the policy is defined for a belief state $p(\mathbf{s})$, the equation for finding the policy is the same as for the MDP on the maximum likelihood state $\operatorname{argmax}_{\mathbf{s}_i} p(\mathbf{s}_i)$.

Another heuristic is the voting heuristic (Simmons and Koenig, 1995). Again, during planning only the underlying MDP is solved. During execution, a distribution over states is maintained, but at every step, every state votes for the action given by the MDP at that state. Each state's vote is weighted by the probability of being in that state as given by the belief distribution. The weighted votes for each action are totalled and the action with the largest total is taken.

The policy for this heuristic is:

$$\pi(p(\mathbf{s})) = \operatorname*{argmax}_a \sum_{i=1}^{N} p(\mathbf{s}_i) \operatorname*{argmax}_a Q(\mathbf{s}_i, a) \tag{5}$$

where $Q(s, a)$ is defined as in equation (4). The assumption made by this heuristic is that although the belief state can represent competing hypotheses, the policy at each of the relevant competing hypotheses is likely to be the same. For example, for a mobile robot travelling down a corridor, even if the mobile robot does not know where it is in the corridor, the optimal policy is the same for all states – continue travelling down the corridor. So long as the uncertainty in the state resolves itself before two high-probability states in the belief require different actions, then this heuristic will approximate the optimal policy, and indeed has been used successfully for mobile robot navigation.

2.2.1 Information Gathering Heuristics Consider the fact that the true value function of the POMDP is piece-wise linear and convex. Recall from section 2.0.2 that the value function is piece-wise linear. The information gathering heuristics make use of a second property of the value function, that it is convex. The value function is highest at the edges of the belief space, where the belief is most narrowly focussed on a single state. In the middle of the belief space, where the probability mass is most evenly distributed between states, the value function is lowest. More intuitively, if the true state of the system is known, it is easy to choose the correct action to maximise the immediate reward. If the true state of the system is however *not* known, then it is comparatively easy to make a mistake, and receive less reward (or even a negative reward) by accident. Therefore, the *expectation* is lower in the middle and higher near the edges of the belief space, which is also a property of convex functions. This is by no means a formal argument, but a proof of the piece-wise linear convexity of the POMDP value function can be found in (Sondik, 1971; Smallwood and Sondik, 1973). A number of heuristics, largely proposed by Littman and Cassandra, attempt to model the effect of actions that push the belief state to the edges of the belief space, or *information gathering* actions.

The first of these is the Q-MDP heuristic (Littman et al., 1995), sometimes called the "Fully-observable after one step" heuristic, allows one step of POMDP value iteration and then uses the Q values for future expected award. The policy for a particular belief $p(\mathbf{s})$ can be written as:

$$\pi(p(\mathbf{s})) = \underset{a}{\operatorname{argmax}} \sum_{i=1}^{N} p(\mathbf{s}_i) Q(\mathbf{s}_i, a) \tag{6}$$

The effect of this policy is to act as if the true state of the problem will become observable after a single step, and as Cassandra notes, "if an action is available which is fairly neutral in terms of rewards", the policy will probably choose that action. If, however, the belief does not resolve itself into a single state after that action then the policy will continue to choose the same action with no forward progress.

Two more heuristics proposed by Cassandra explicitly address the need to explicitly model information-gathering actions, and to trade off the information against reward (Cassandra et al., 1996). These are dual-mode controllers that alternate between pursuing expected reward and gathering information to refine the belief state. The entropy of the belief state is given by

$$H(p(\mathbf{s})) = -\sum_{i=1}^{N} p(\mathbf{s}) \log_2 p(\mathbf{s}) \tag{7}$$

$$\overline{H}(p(\mathbf{s})) = \frac{H(p(\mathbf{s}))}{H(u)} \tag{8}$$

where $H(u)$ is the entropy of the uniform distribution over all states, and $\overline{H}(p(\mathbf{s}))$ is the entropy normalised to lie in the interval $[0, 1]$. The first of heuristic is the entropy control that gives the policy:

$$\pi(p(\mathbf{s})) = \begin{cases} \operatorname{argmin}_a E_z \left[H(p(s|a, z)) \right] & \text{if } \overline{H}(p(\mathbf{s})) > \kappa \\ \pi_X(p(\mathbf{s})) & \text{otherwise} \end{cases} \tag{9}$$

where $\pi_X(p(\mathbf{s}))$ is some other heuristic, such as the most-likely state heuristic. This heuristic simply prevents the entropy, or uncertainty about the belief state, growing larger than the arbitrary parameter κ, and can act suboptimally when the uncertainty is irrelevant.

The improved version of this heuristic is the Weighted-Entropy heuristic (Cassandra et al., 1996). Given a value function for the completely-unobservable MDP $V_{CU}(\mathbf{s})$ and the value function for the completely-observable underlying MDP $V_{CO}(\mathbf{s})$, the policy given by this heuristic is given by the Q-value:

$$\pi(p(s)) = \underset{a}{\operatorname{argmax}} \left[\overline{H}(p(\mathbf{s}))^{\kappa} \sum_{i=1}^{N} p(\mathbf{s}_i) Q_{CU}(\mathbf{s}_i, a) + (1 - \overline{H}(p(\mathbf{s}))^{\kappa}) \sum_{i=1}^{N} p(\mathbf{s}_i) Q_{CO}(\mathbf{s}_i, a) \right] \tag{10}$$

Essentially, this method arbitrates between the action dictated by a completely-observable model (V_{CO}) which choose actions as if the system knows the state of the world exactly, and the action dictated by a completely-unobservable

model (V_{CU}), which will choose actions as if the system has no idea where it is. The two methods vote for actions, and the voting is weighted by the current entropy. The higher the entropy, the higher the weight on acting as if the system has no knowledge of the state of the world. The advantage to this method is that if the same sequence of actions leads to the same expected reward regardless of the current belief state, the heuristic will take advantage of such a sequence. Such sequences might exist in worlds with funnelling actions or funnelling states. However, if no such sequences exist, then this heuristic still embodies at every step a choice between state identification, and pursuing expected reward. Furthermore, if funnelling actions or states exist for *some* belief states but not others, this heuristic cannot take advantage of such actions. It should be noted that Cassandra proposes versions of both heuristics that compute the entropy of the action space rather than the belief space, which can be viewed as combining both the entropy-based heuristics with the voting heuristic discussed above.

2.3. Failures of Heuristics

There are two reasons for the failures of the heuristics. The heuristics in general overestimate the ability to discover the true state of the system or overestimate the need to discover the true state. Furthermore, the greedy nature of the heuristics is a failing that needs to be addressed.

The first two heuristics (maximum-likelihood state and the voting methods) rely at heart on the value function of the underlying MDP. The convexity of the POMDP value function suggests that periodically, the POMDP policy should take an action that may have low immediate reward, but moves the belief state away from the middle of the belief space. Because the underlying MDP is by definition fully observable, it cannot accurately represent the effect of information gathering actions that refine the belief state. To the MDP, and consequently the heuristics, an information gathering action is a wasted action, unless this action also optimally moves the agent closer to the goal. The Q-MDP heuristic is a step towards addressing the problem, by acknowledging that the true underlying state may not always be known, but makes the assumption that by waiting long enough, the true state will become known.

Conversely, the entropy and weighted-entropy heuristics explicitly model the problems with uncertainty about the true state. However, both heuristics over-compensate by modelling the fact that uncertainty above an arbitrary level is never acceptable (entropy) or is acceptable only if the expected reward is the same regardless of belief state (weighted entropy).

Finally, none of the heuristics is capable of using an arbitrarily long sequence of actions that will refine the belief state. If no single action can lead to identifying the true state, then all of these heuristics are likely to fail. However, the failures of these heuristics provide motivation for the algorithm I propose in this thesis.

3. The Augmented MDP

The approach proposed here is to address the problem of capturing the full effect of information gathering actions and in particular sequences of information gathering actions, while avoiding the complexity of representing the full belief state during planning. Two motivating examples will be robot navigation and speech dialogue management.

Conventional planning suffers from an inability to capture many kinds of uncertainty about real world processes. As we have seen, the major failing of the greedy heuristics is that they fail to capture the idea of planning to refine the belief; however, representing the belief explicitly leads to high computational costs of solving full POMDPs. Instead, we wish an intermediate solution that can plan to gather information while maintaining a degree of tractability. This forms a kind of continuum as depicted in figure 1, with the proposed solution somewhere in the middle.

Many problem domains have specific kinds of beliefs; for these domains it is unnecessary to generate a policy for every possible belief state. Furthermore, the kind of beliefs that are likely in such domains can be represented compactly, or summarised, by a sufficient statistic. For example, if the belief state is known to be a uni-modal Gaussian with a single variance, then the sample mean and variance are jointly sufficient statistics for recovering the true model. Consequently, the belief state can be represented using a two-dimensional vector regardless of the number of states in the underlying MDP, and full planning can occur without loss of generality.

Choosing the appropriate sufficient statistic is not trivial, however, and is a function of the particular problem

domain and the belief states that arise in it. Throughout the remainder of this work, we will be assuming that the maximum-likelihood state (MLS) $\operatorname{argmax}_{\mathbf{s}} p(\mathbf{s})$, and the distribution entropy form a set of jointly sufficient statistics.

Using the entropy for part of the belief state representation is a reasonable choice, because in some sense it captures the uncertainty of the belief state. Recall that the convexity of the value function dictates that the expected reward will be lower towards the center of the belief space. The entropy of the belief state corresponds to the distance the belief state is from the edges of the belief space. The higher the entropy, the closer to the middle of the belief space the system is in, and the lower the subsequent expected reward. By modelling this uncertainty explicitly with entropy, we allow the planner to detect readily when the uncertainty is too high, and take information gathering actions, pushing the belief state back towards the edges of the belief space.

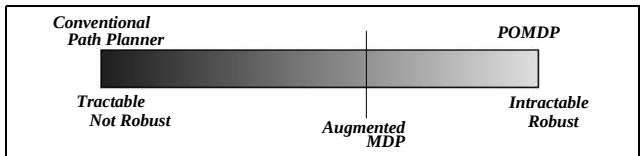


Figure 1: The augmented MDP planner lies somewhere on a continuum between the deterministic planners and the full-blown POMDP solution.

We will show that the MLS-entropy pair, while not true jointly sufficient statistics, are reasonable choices for the problem domains we are addressing. It is this state representation: the MLS augmented by the entropy that we refer to as the augmented MDP. We next show how we can take an arbitrary POMDP and solve it using MDP techniques using the augmented representation.

3.1. Planning with the Augmented MDP

For planning with the augmented MDP, we represent the belief state as the tuple

$$\mathbf{b} \cong < \operatorname*{argmax}_{\mathbf{s}} p(\mathbf{s}); H(p(\mathbf{s})) > \tag{11}$$

We concatenate the MLS with the entropy; this notation should not be confused with the inner product. We are approaching a conventional state representation, such as one that could be used by an MDP solver. However, we still have continuous state factors. We therefore discretise the underlying state representation if it is not already discrete, and we discretise the possible entropy levels $H(p(\mathbf{s}))$, to give an entirely discrete and finite state space[2].

This discrete and finite representation allows us to use value iteration over the belief state exactly as we did for solving MDPs in equation (1). However, for a given POMDP specification, the transition probabilities for the augmented MDP, $T_H(\mathbf{s}', a, \mathbf{s})$ have to be reconstructed for the new state representation, and are a function of both T and O, the emission probabilities.

$$\begin{aligned} T_H(\mathbf{b}_i, a, \mathbf{b}_j) &= p(\mathbf{b}_j|\mathbf{b}_i, a) && (12)\\ &= p(p_j(\mathbf{s})|p_i(\mathbf{s}), a) && (13)\\ &= \alpha \sum_{\mathbf{s}'} p(z|\mathbf{s}')p(\mathbf{s}'|a) && (14)\\ \text{for } z \text{ such that } b_j \approx p_j(\mathbf{s}) &= \alpha p(z|\mathbf{s}') \sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s}, a)p_i(\mathbf{s}) && (15) \end{aligned}$$

Computing equation (13) is difficult (but not impossible), because it requires inverting the forward propagation of belief. Much easier, we instead propagate the belief forward to identify a possible posterior state and its transition probability at the same time. Given an initial belief $\mathbf{b}_i$, an action a and an observation z, we first generate a posterior belief $\mathbf{b}_j$ in the following manner:

$$\begin{aligned} \mathbf{b}_j \approx p(\mathbf{s}'|z, a) &= \alpha p(z|\mathbf{s}')p(\mathbf{s}'|a)\\ &= \alpha p(z|\mathbf{s}') \sum_{i=1}^{N} p(\mathbf{s}'|\mathbf{s}_i, a)p(\mathbf{s}_i) && (16) \end{aligned}$$

[2]Entropy in general is not bounded in the same way that total probability is, but for a fixed and finite number of states entropy is bounded with maximum $H_{max} = -\frac{\log 1/|S|}{|S|}$ where $|S|$ is the size of the state space.

and then compute the transition probability for as a function of that given z as:

$$\begin{aligned} T_H(\mathbf{b}_i, a, \mathbf{b}_j)) &= \alpha p(z|a, \mathbf{s}) \\ &= \alpha \sum_{i=1}^{N} p(z|\mathbf{s'}_i) \sum_{j=1}^{N} p(\mathbf{s'}_i|\mathbf{s}_j, a) p(\mathbf{s}_j) \end{aligned} \tag{17}$$

In equation (17), we normalise the transition probabilities by a constant α so that

$$\sum_{j=1}^{N} T_H(\mathbf{b}_i, a, \mathbf{b}_j) = 1 \tag{18}$$

The above equations use summation instead of integration over beliefs because the belief space is discretised.

It may seem odd to be computing probabilities of final belief states; typically, transitions between belief states are considered to be deterministic. However, because the posterior belief depends not only on the action but also the observation, and the observation is emitted stochastically, there is a probability distribution over the posterior beliefs as a function of observations that must be captured by the transition function $T_H(\mathbf{b}_i, a, \mathbf{b}_j)$.

In the preceding description, we are also not computing explicitly the transition function for all pairs of belief states. In fact, in most problems we sample from the set of possible observations, according to the emission probabilities given by O, to get a set of likely posteriors. All other transition probabilities we set to zero, and then normalise by α. This normalisation step is not needed in general, if we explicitly compute the transition probabilities for all posterior beliefs.

Finally, we need to modify the reward function, $R(\mathbf{s}, a, o)$. For the domains we have considered, the reward function is only a function of the state, $R : \mathcal{S} \mapsto \Re$. In order to extend the reward to the augmented MDP formalism, we apply *expected* immediate reward:

$$\begin{aligned} R(\mathbf{b}) &= \mathrm{E}_{\mathbf{b}}(R(\mathbf{s})) \\ &= \sum_{i=1}^{N} R(\mathbf{s_i}) p(\mathbf{s_i}) \end{aligned} \tag{19}$$

This has the effect of punishing the planner for producing a plan that arrives at the goal with high entropy, unless the maximum reward can be achieved across states that are covered by a high entropy belief state.

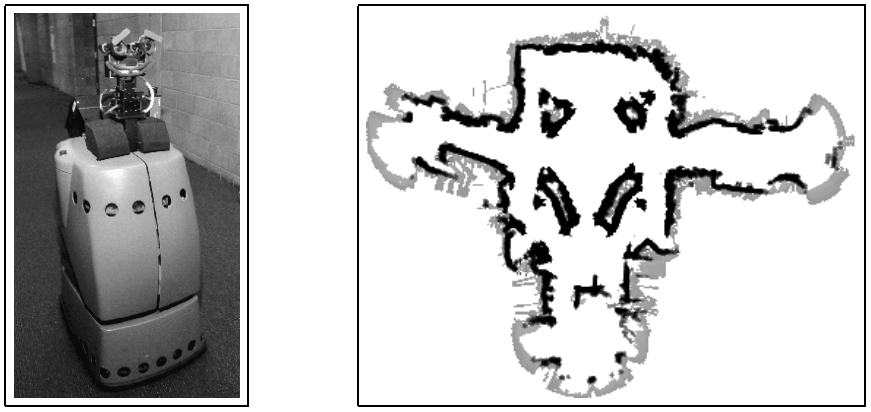
4. Preliminary Results

With the machinery described in the preceding section, it is then a simple matter to apply Bellman's equation and produce an optimal plan for belief state representation given in equation (11). I shall give two examples of problem domains in which the augmented MDP has been applied, and evaluate the success of our approach.

4.1. Mobile Robot Navigation — Coastal Navigation

The first example of the augmented MDP is motivated by Minerva, the museum tour-guide robot that operated in the Smithsonian for two weeks in the summer of 1998. As discussed earlier, a mobile robot like Minerva must be able to navigate reliably without becoming lost. Early motion planners assumed that a robot would never be lost – that a robot could always know its position via dead reckoning without error (Latombe, 1991). This assumption proved to be untenable due to the small and inevitable inconsistencies in actual robot motion; robots that rely solely on dead reckoning for their position estimates lose their position quickly.

Mobile robots now perform position tracking using a combination of sensor data and odometry (Burgard et al., 1996; Sim and Dudek, 1998). In particular, Minerva used a laser range sensor and a prior map in conjunction with Markov localisation to maintain knowledge of its pose. Figure 2 shows an overhead map of the environment. The



(a) Minerva (b) The Museum

Figure 2: In panel (a), Minerva, the museum tour-guide robot that operated in the Smithsonian National Museum of American History for 2 weeks in the summer of 1998. In panel (b), an overhead map of the National Museum of American History.

grey-black areas are walls or display cases, and the white area is free space. The dimensions of the museum are 53m by 67m.

Unfortunately, Minerva's localisation method suffered from failures in the museum. This was as a result of two factors. The first was the sparseness of the environment. The robot's operational area of the museum consisted of large areas of free space, with very little environmental structure, making the localisation problem hard. Furthermore, the environment was extremely crowded, which further confounded the problem because the surrounding crowds of people blocked the laser range sensor. The crowds both preventing the robot from seeing the true structure of the environment which was contained in the map, and also introducing false structure into the sensor perception.

We developed a navigation algorithm that took into account both the local structure of the environment for localisation, and the probability of the structure being useful or blocked by crowds[3]. By modelling the effect of sensing at every point in space, and in particular the effects on the localisation process of both the local environmental structure and the local crowds, we were able to use the augmented MDP machinery to generate more robust trajectories. These more robust trajectories minimised the possibility of becoming lost due to either crowds or uninformative sensor data.

The problem of measuring uncertainty with entropy and then acting to improve the localisation estimate is not a new one, and can be found in the active localisation literature (Fox et al., 1998). However, active localisation does not plan trajectories that improve localisation while achieving some goal, but actually dictates the optimal trajectory for localisation. The augmented MDP can be viewed as a compromise between active localisation and conventional distance-optimal planning.

Similarly, POMDP-style planners have been used to recover from localisation failure (Nourbakhsh et al., 1995; Koenig and Simmons, 1996; Takeda et al., 1994), but the greedy nature of the heuristics these planners use prevents the global style of planning that the augmented MDP generates. In topological environments, full POMDP planners have been used with some success (Mahadevan, 1998). However, these approaches rapidly approach the complexity limitations of full POMDP planners and are not tractable for the richer grid-based representation presented here.

[3]The original navigation algorithm for dealing with the problems in the museum was called "Coastal Navigation" (Roy et al., 1999). This algorithm assigned a cost to being uncertain along the trajectory of the robot from point to point as well as the travel cost of passing through that point, and then it attempted to minimise the overall cost of the trajectory. However, the coastal navigation algorithm proved to be overly limited. By modelling uncertainty as a cost, the uncertainty could only monotonically increase. It was not possible to model arbitrary decreases in positional uncertainty due to periodic re-localisation. Coastal navigation forced the robot to stay as localised as possible as often as possible, which was not always the ideal behaviour.

4.1.1 The Augmented MDP for Navigation We first summarise Markov localisation process (Burgard et al., 1996), and the demonstrate how to convert the conventional path planning problem into an augmented MDP that models positional uncertainty, in order to minimise the probability of becoming lost.

Let the robot's position be given by the initial probability distribution, $p_{\mathbf{x}}$, defined over $\mathbf{X} = (X, Y, \Theta)$. We will represent the belief state as

$$p(x, y, \theta) = \langle \underset{x,y,\theta}{\operatorname{argmax}}\, p(x, y, \theta); H(p(x, y, \theta)) \rangle \tag{20}$$

We assume a laser range finder that projects a beam of infrared light parallel to the floor and about 30cm above it. This sensor returns 180 measurements where each measurement corresponds to the nearest obstacle along a particular direction, for 180 directions around one half of the circumference of the robot, in 1° increments. We in practice have two such sensors back to back, which provides a 360° field of view for the sensor.

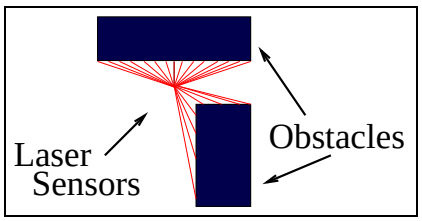


Figure 3: An example laser range measurement.

Localisation consists of a prediction step, from taking an action a, and then an update step from taking a sensor measurement $\mathbf{z}$. The prediction step is given as:

$$\begin{aligned} p(\mathbf{x}'|a) &= \int_{\mathbf{X}} p(\mathbf{x}'|\mathbf{x}, a) p(\mathbf{x}) d\mathbf{x} \\ &= \sum_{\mathbf{X}} p(\mathbf{x}'|\mathbf{x}, a) p(\mathbf{x}) \end{aligned} \tag{21}$$

where $p(\mathbf{x}'|\mathbf{x}, a)$ is the motion model learned from the robot. The true prediction step is an integral because the robot motion is continuous, but we discretise the space into a grid world, allowing the summation approximation in equation (21).

The robot acquires a sensor measurement $\mathbf{z}$, for example a set of range data from a laser sensor. The update step takes $p(\mathbf{x}|a)$ and the sensor data, $\mathbf{z}$, and returns the posterior probability distribution $p(\mathbf{x}|\,\mathbf{z})$, again defined over the space of poses of the robot, (X, Y, Θ).

The probability $p(\mathbf{x}|\,\mathbf{z})$ is given as:

$$p(\mathbf{x}|\,\mathbf{z}) = \frac{p(\mathbf{z}|\,\mathbf{x}) p(\mathbf{x})}{p(\mathbf{z})} \tag{22}$$

where $p(\mathbf{x})$ is the position distribution after the prediction step, and $p(\mathbf{z}|\,\mathbf{x})$ is computed from the sensor model and the robot's map of the environment.

The term $p(\mathbf{z})$ is the likelihood of observing sensor data $\mathbf{z}$, and is computed from the prior position distribution, the sensor model and the environmental map.

$$p(\mathbf{z}) = \sum_{\mathbf{X}} p(\mathbf{z}|\,\mathbf{x}) p(\mathbf{x}) \tag{23}$$

As above, although equation (23) should be an integral, the grid world allows us to convert the integral into a summation.

4.1.3 Modelling Dynamic Obstacles Recall that the stated goal of using the augmented MDP was to generate trajectories that avoided areas of sparse environmental structure and crowdedness. The first idea, of modelling areas of sparse environmental structure, is captured fully by Markov localisation. The effect of sensing in a particular location is modelled by equation (22); if the environment does not contain enough sensor cues for accurate localisation, the

output of this equation will be a wide probability distribution. The entropy will be high, and the entropy of subsequent belief states is also likely to be high, leading to a poor expected reward at the goal. Conversely, if the output of equation (22) is a sharp probability distribution, then future belief states should reflect this fact and the expected reward at the goal will be high.

However, Markov localisation as described has no concept of dynamic obstacles, or people, and the effect they have on sensing and the subsequent posterior beliefs. Modelling the effect of dynamic obstacles is of course a sensor-dependent process, but we restrict our development to the laser range sensors that we are using in this particular domain.

Recall that each sensor measurement $\mathbf{z}$ is composed of 180 individual ranges z_i. If we assume that people are uniformly distributed over the environment, then we can model the likelihood that any single laser range, z_i, is corrupted by a person blocking the range; we can model this probability as a geometric distribution:

$$p^{corrupt}(z_i) = 1 - \gamma^{\|z_i\|} \tag{24}$$

$z_i : 0 \leq \|s\| \leq MaxRange$ is the ith range measurement in the measurement z, $\|z_i\|$ is the actual range and $\gamma : 0 \leq \gamma \leq 1$ is the probability that any particular point in the environment is occupied by a dynamic obstacle: for the case of museum, this is simply the estimated number of people in the museum, divided by the area of the free space of the museum. Figure 4 shows the probability of corruption, $p^{corrupt}$ as a function of measurement distance.

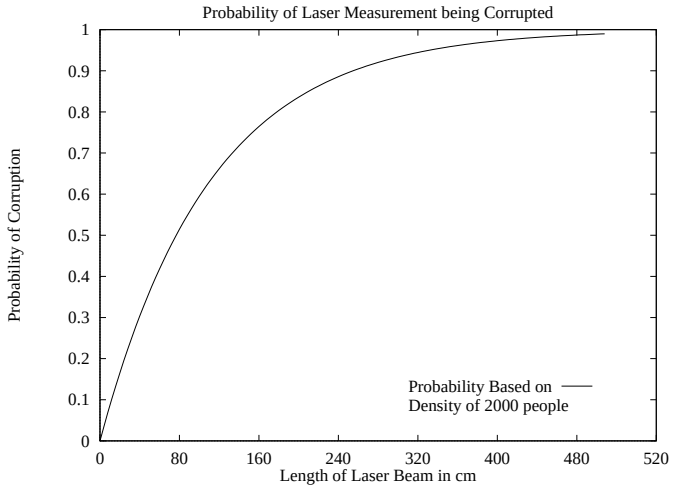


Figure 4: Probability of corruption of laser measurement, as a function of measured distance, given 2000 people in the museum.

In order to account for this corruption model, we need to alter the effect that individual sensor measurements have on the posterior belief state. We make a simplifying assumption that each component of the sensor measurement $z_i(x, y) : \mathbf{z}(x, y) = \{z_1(x, y), \ldots, z_n(x, y)\}$ is independent. This allows us to compute the expected value of the belief state $p(\mathbf{x}|z, a)$ by averaging over the posterior belief state of each component of the measurement, weighted by the probability that the component is corrupted as in equation (24). We make this independence assumption for the sake of tractability[4]. We can represent this process as:

$$p(\mathbf{x}|\mathbf{z}) = \sum_{i=1}^{K} \frac{p(z_i|\,\mathbf{x})p(\mathbf{x})}{p(z_i)} p^{corrupt}(z_i) \tag{25}$$

In practice we only perform the averaging after the belief state has been compressed to its augmented representation as given in equation (11), in the interests of speed.

[4]In reality, the range to an obstacle at direction θ_i is highly correlated with the range θ_{i+1} in most environments. However, we can make reasonable conclusions about the posterior belief, and this method has the advantage of being computationally fast.

4.1.4 Transitions using Markov Localisation We have a probabilistic sensor model and a map of the environment, we can determine the probability of a sensor measurement $p(\mathbf{z}|\mathbf{x})$. We also have a probabilistic model of the robot motion, $p(\mathbf{x}'|\mathbf{x}, a)$. Both the motion model and the sensor model were acquired empirically. Using these models, we can compute the transition probability between a prior belief state $p(\mathbf{x})$ and a posterior belief state $p(\mathbf{x}')$.

As discussed in section 3.1, we do not in fact compute the transition probability explicitly, but instead generate a posterior belief state for action-observation pairs, and compute the probability of each forward transition. In the grid-based world there are only 4 possible actions $\{north,\ south,\ east,\ west\}$ (or if a robot-centric reference frame is preferred $\{left,\ right,\ forward,\ backward\}$). However, there are 700^{360} possible sensor measurements[5], and it is neither computationally feasible nor worthwhile computing a transition probability for each possible belief that would result from each of these possible observations, since most are extremely unlikely. Instead, we sample from the likelihood distribution Z of the observations at the most likely state:

$$Z = p(z|\mathbf{x}_{ml}) \tag{26}$$

$$\mathbf{x}_{ml} = \operatorname*{argmax}_{\mathbf{x}} \sum_{\mathbf{x}'} p(\mathbf{x}'|\mathbf{x}, a)p(\mathbf{x}) \tag{27}$$

We sample the 9 maximum likelihood observations from each of the possible (x, y)[6] positions in the 3x3 grid centred around the maximum likelihood position after the prediction phase given in equation (21). It should be noted that the decision to sample in this manner may be problematic. The samples are no longer independent, and 9 samples may not be enough to accurately represent the transition probabilities. However, this sampling mechanism seems to work in practice.

For each of the 9 sample observations, we compute a posterior belief $\mathbf{b}_j = p(\mathbf{x}'|z, a)$, and use this to generate the transition probabilities, as in equation (17).

More succinctly, given an initial belief $\mathbf{b}_i$, an action a and one of the sampled observations z, we get a subsequent belief $\mathbf{b}_j$ from equations (21), (22 and (25). We get this belief with probability:

$$\begin{aligned} T_H(\mathbf{b}_i, a, \mathbf{b}_j) &= \alpha p(z|a, \mathbf{x}) \\ &= \alpha \sum_{i=1}^{N} p(z|\mathbf{s'}_i) \sum_{j=1}^{N} p(\mathbf{s'}_i|\mathbf{s}_j, a)p(\mathbf{s}_j) \end{aligned} \tag{28}$$

Note that $p(\mathbf{x}|a)$ is for a single state $\mathbf{x}$, so this is a single probability value, not a distribution. Again, as in equation (18), α normalises the transition probabilities for a given belief-state action pair so that the probabilities sum to 1.

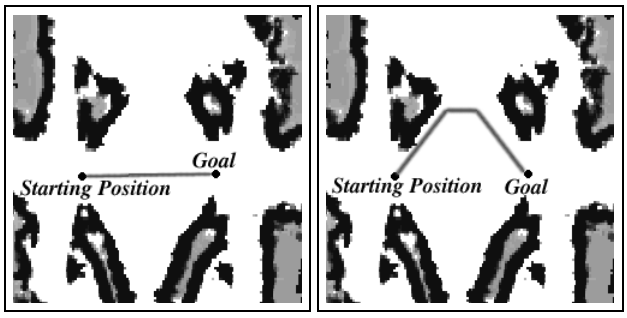
We can therefore compute the transition probabilities as follows:

- For all possible beliefs $< \mathbf{x_i}; H(p(\mathbf{x}_i)) >$:
  - For all actions a_j:
    1. Reconstruct the full belief state from the compressed representation
    2. Compute $p(\mathbf{x}_i|a_j)$ from equation (21)
    3. Generate potential observation z_k from equation (26)
    4. Generate new posterior $p(\mathbf{x'}_i|z_k, a_j)$ from equation (25)
    5. Compute transition probability $T_H(p(\mathbf{x'}_i), a_j, p(\mathbf{x}_i))$ from equation (28)
    6. Repeat for all 9 sampled observations

[5]The sensor takes 360 such measurements in 1° increments around the robot, and each range measurement is 1 cm resolution from 1m to 8m, giving 700 possible such range measurements.

[6]The sensor is rotationally invariant, so the sensor measurement for a given (x, y) is the same regardless of the orientation.

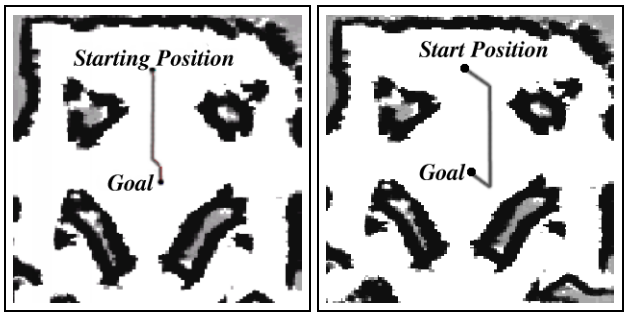
4.1.5 Experimental Results We tested the augmented MDP navigation in the Smithsonian museum (Roy and Thrun, 1999). Figure 5 shows the effects of two different planners in the sample environment. Panel (a) shows the trajectory of a conventional, shortest path planner. Note that the robot moves directly towards the goal. Panel (b) shows the trajectory given by the augmented MDP planner. Panel (c) shows the sensor map of the environment. The black areas show obstacles and walls, and the light grey areas are where no information is available to the sensors, because all environmental features are outside the range of the sensors. The dark grey areas indicate areas where the information gain from the sensors is *not* zero; the darker grey the area, the better the information gain from the sensors. Notice that the robot sacrifices a shorter path, for maintaining higher positional knowledge throughout the trajectory. The higher positional knowledge comes from being closer to the walls of the display cases, increasing the ability to localise because more environmental structure is in view, and also because the likelihood that the information from the sensors is corrupted by crowds is reduced.



(a) Conventional trajectory (b) Augmented MDP trajectory

Figure 5: An example of a conventional trajectory (a) and the augmented MDP trajectory (b) for the same start and goal point.

Figure 6 shows a different example also in the museum. Again in panel (a) is the conventional trajectory and panel (b) is the trajectory from the augmented MDP. This example illustrates the ability of the planner to take advantage of information gathering actions. The robot moves towards the goal, goes past it, relocalises once it is in sensor range of the obstacle, and then returns to the goal. These periodic relocalisations are essential for the robot to arrive at the goal with minimum positional uncertainty, and maximum reliability.



(a) Conventional trajectory (b) Augmented MDP trajectory

Figure 6: Another example of a conventional trajectory (a) and the augmented MDP trajectory (b) for the same start and goal point.

We performed quantitative analysis over the performance of the augmented MDP compared to the conventional planner. In order to show that the augmented MDP really does improve navigation reliability, we decreased the

maximum sensor range; this substantially hindered the ability of the robot to gather information and localise accurately. As a measure of navigation reliability, we examined the entropy of the final belief state at the goal, over a series of simulated trajectories in the museum environment. The lower the entropy of the belief state, the better the robot's positional knowledge and consequently the more reliable the goal attainment. Figure 7 depicts the average positional certainty at the goal of the augmented MDP compared to the conventional planner, as a function of maximum laser range, from 0.5m to 5m. The upper line is the performance of a conventional shortest-distance path planner, and the lower line is the coastal planner. The coastal planner has a lower uncertainty for all ranges of the laser sensor, and is substantially lower at shorter ranges, confirming that the augmented MDP has the most effect when the localisation is worst. Even better, the augmented MDP does not appear to degrade substantially when the localisation is at its worst.

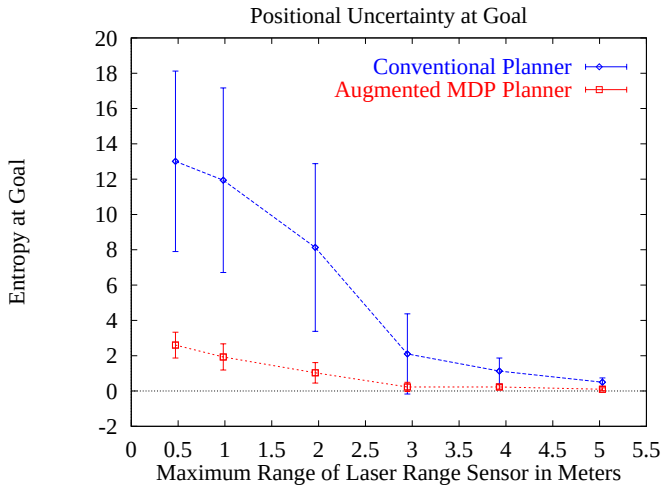


Figure 7: The performance of the augmented MDP compared to the conventional motion planner, in the face of decreasing sensor ability. As the sensor becomes worse and worse (the left-hand side of the graph), the augmented MDP's performance does not degrade nearly as quickly as the conventional planner.

4.2. Speech Dialogue Management

The development of automatic speech recognition has made possible more natural human-computer interaction, and more recently, rich interaction between humans and robots (Torrance, 1994; Westphal and Waibel, 1999; Denecke and Waibel, 1997). Speech recognition and speech understanding, however, are not yet at the point where a computer can reliably extract the intended meaning from every human utterance. Recent research in dialogue management has shown that Markov Decision Processes can be useful for generating policies (Young, 1990; Levin et al., 1998); the actions are the speech productions from the system, the state is represented by the dialogue as a whole, and the goal is to maximise some reward, such as fulfilling some request of the user. However, the correct way to represent the state of the dialogue is still an open problem (Singh et al., 1999). A common solution is to restrict the system to a single goal, for example booking a flight in an automated travel agent system; the system state is described in terms of how close the agent is to being able to book the flight.

Figure 8: Florence Nightingale, the nursebot used as the robot platform for the experiments in speech dialogue management.

Such systems suffer from three principal problems. Firstly, most existing dialogue systems are built for a single purpose over a single task domain, for example retrieving e-mail or making travel arrangements (Levin et al., 1998). While it is not impossible to extend these systems to multiple domains, most existing dialogue systems do not model confidences on the human utterances, and therefore do not account for the reliability of utterances while performing a strategy. Some systems do use the log-likelihood values for speech utterances, however, these values are only thresholded as to whether or not the utterance needed confirmation or not (Niimi and Kobayashi, 1996; Singh et al., 1999). Secondly, system-initiated strategies perform best with these approaches, however mixed-initiative strategies are a more natural model for human-robot interaction. Finally, the interaction between different task domains is difficult to model.

The real problem that lies at the heart of these three issues is that of observability. The ultimate goal of a dialogue system is to satisfy a user request; however, what the user wants is only partially observable at best. A conventional MDP demands to know the current state at all times, therefore the state has to be wholly contained in the system. System-initiated dialogues are successful because there is no uncertainty in determining when the user wants something, and a single task-domain system works well, because the relevant goals and actions are easily specified.

4.2.1 Dialogue Systems and POMDPs We consider a POMDP to be a natural way of modelling dialogue processes when the state of the system is viewed as the state of the user, inverting the conventional notion of state in a dialogue. The uncertainty model inherent in a POMDP policy allows the dialogue planner to recover from noisy or ambiguous utterances in a natural and autonomous way. At no time does the machine interpreter have any direct knowledge of the state of the user, i.e, what the user wants. The machine interpreter can only infer this state from the (noisy) speech of the user.

The particular domain that we use in the following discussion is based on a mobile robot, Florence Nightingale (Flo), developed as a prototype nursing home assistant. Flo uses the Sphinx II speech recognition system (Ravishankar, 1996), and the Festival speech synthesis system (Black et al., 1999). Figure 8 shows a picture of the robot.

Since the robot is a nursing home assistant, we use task domains that are relevant to everyday life. Table 1 shows a list of the task domains the user can ask about: the time, the weather, what is on different TV stations, and can also ask to deliver a video message (called *facemail*) to another person. These abilities have all been implemented on Flo, and the information about the weather and TV schedules is downloaded on request from the web.

From the domains of conversation, we extracted 17 states, that correspond to possible intentions of the user. The state representation was compiled by hand, and does not necessarily correspond to the true state representation of possible user intentions. A future experiment is to validate the state representation with a set of user experiments.

If we translate these tasks into the MDP framework that we have described, the decision problem has 17 states, and the state transition graph is given in figure 9. The different tasks have varying levels of complexity, from simply saying

Time
Weather (Current and for tomorrow)
TV Schedules for different channels (ABC, NBC, CBS)
Delivering video messages (facemail)

Table 1: The task domains for Flo.

`no_request`	`want_tv`	`want_facemail`
`request_begun`	`want_nbc`	`recording_facemail`
`request_done`	`want_abc`	`want_facemail_sent`
`want_time`	`want_cbs`	`want_facemail_mike_m`
`want_weather`	`want_current_weather`	`want_facemail_mike_v`
	`want_tomorrow_weather`	`want_facemail_sebastian`

Table 2: The list of states used in the full decision problem in these experiments.

the time, to delivering facemail, in which the user moves through four different states before the task is completed.

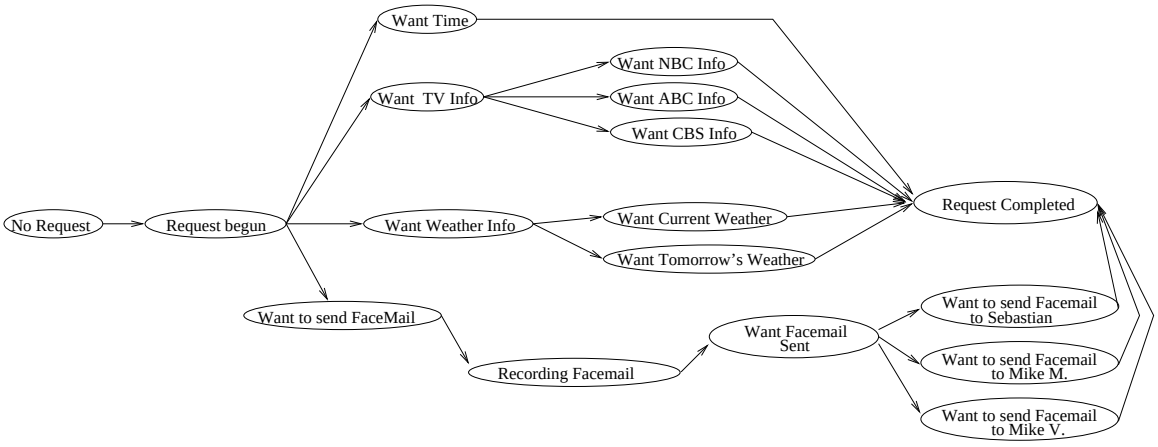


Figure 9: The 17-state MDP model, representing the kinds of requests Flo can handle.

We compiled this into a POMDP model with 17 states, 22 actions and 18 observations. We converted this POMDP into an augmented MDP using the equations given in section 3. Once again, we represent the belief $\mathbf{b}$ as:

$$\mathbf{b} =< \underset{s}{\operatorname{argmax}}\, p(s); H(p(s)) > \tag{29}$$

Similar to robot navigation, we maintain the belief state through a two-stage prediction-update cycle. The prediction phase as a result of taking an action a (usually a spoken response to the user in this domain) is given by:

$$p(s'|a) = \sum_{i=1}^{N} p(s'|a, s_i)p(s_i) \tag{30}$$

where $p(s'|a, s_i)$ is given by the model. For robot navigation, this action model was learned empirically from actual motion by the robot and models the non-deterministic nature of the robot's actual motion in space. In the spoken dialogue case, the action model was hand-coded to model the user's non-deterministic change in intentions; for example, even if the user asked initially for the time, the user may change their mind and ask for the weather even if the system

complies with the initial request. The action model should in fact be learned or at least verified by a user study, but for the purposes of this demonstration of the augmented MDP, the simple hand-coded model suffices.

Finally, we update the belief by applying the effect of receiving an utterance from the user z:

$$p(s|z) = \alpha p(z|s)p(s) \tag{31}$$

where $p(z|s)$ is given by the model. Just as the action model was hand-coded, we hand-coded an observation emission model that reflected the likely errors in the speech recognition system.

Finally, the reward is structured such that taking an incorrect action (such as `say_hello` when no greeting has been given, or `deliver_mail` when no mail has been given) had a reward of -10, but the reward for taking the correct action is +20. Some special cases exist: the reward for transitioning to the `request_done` state is +200, the reward for asking a question or confirming a response is -1, and the reward for delivering a message to the wrong person is -200. The reason for the additional negative reward in delivering to the wrong person is to highlight the difference between a costly action that is merely irritating (giving an inappropriate response) and an action that can be much more serious (having the robot leave the room at the wrong time, or travel to the wrong destination).

4.2.2 Planning and Executing in the Speech Domain Given the model as specified, we convert the POMDP into the augmented MDP as in equations (11) and (17), by representing the belief as the *(maximum-likelihood state, entropy)* pair, and computing the transition probabilities appropriately. Because the set of observations is very small, it is not necessary to sample from a distribution of observations. We compute the transition probabilities for all destination beliefs.

However, during execution, the actual execution of the policy is slightly different. Although policy generation assumes the observation emission model as given by $\mathcal{O}$, we use a different emission model. Sphinx, the speech recognition system provides estimates of the probabilities of each possible utterance at run-time. These estimates rarely (if ever) reflect the model probabilities, however, they are used to update the posterior belief of the system more accurately than the prior model would. In this way, it may be that the policy computed from the model may be optimal over the course of several dialogues (assuming that the planning model parameters in $\mathcal{O}$ are the maximum-likelihood parameters over the course of several dialogues), but the policy may be sub-optimal over an arbitrary single dialogue. The best way to proceed with dynamic emission probabilities is not obvious, but it seems likely that dynamic emission probabilities are representative of real world processes such as speech recognition.

Table 3 shows an example dialogue from the augmented MDP system. The left hand columns are the emitted observation and the entropy of the system that resulted. The most interesting lines are the fourth and fifth lines of the dialogue: although the true state of the user is `want_facemail`, the observation made is that the user is in state `want_time`. The conventional MDP would take the observation as given, but the observation from the speech recognition system was highly uncertain, so the entropy of the belief becomes quite high (20). The entropy stays high before getting a more certain observation. The system confirms the desired state, before moving on. We see a similar action a little later; here, because the cost of mis-delivering a facemail is very high, the system takes confirmation action even when the entropy is relatively low.

Observation	Belief Entropy	Action	Reward	True State
<silence>	0	do_nothing	20	none
flo hello	2	ask_repeat	-1	request_begun
flo hello	0	say_hello	20	request_begun
was on a	20	ask_repeat	-1	want_facemail
flo time is	20	ask_repeat	-1	want_facemail
flo i want to deliver please	6	confirm_facemail	-1	want_facemail
flo i want to deliver facemail	0	say_begin_recording	20	want_facemail
<recording_facemail>	0	record_facemail	20	recording_facemail
flo send it	1	ask_repeat	-1	want_facemail
flo sent it	0	ask_which_facemail	20	want_facemail
flo i want to send it to mike	3	ask_repeat	-1	want_facemail_mike_m
flo i want to send it to mike m	0	deliver_facemail_mike_m	200	want_facemail_mike_m
thanks flo goodbye	0	do_nothing	20	request_done

Table 3: An example dialogue, from the augmented MDP.

4.3. Simulated Experimental Results

We examined the performance of the augmented MDP system as compared to a full POMDP solution and an MDP dialogue manager. The full POMDP solver we used was provided by Cassandra and used the Incremental Improvement algorithm (Cassandra et al., 1997). The full POMDP solver was unable to find a policy for more than 7 states, so the first result compares the dialogue managers on a restricted domain of only 7 states. The experiments consisted of generating observations synthetically, using a user model process. The observations were generated according to the prior observation model, and fed to the dialogue manager.

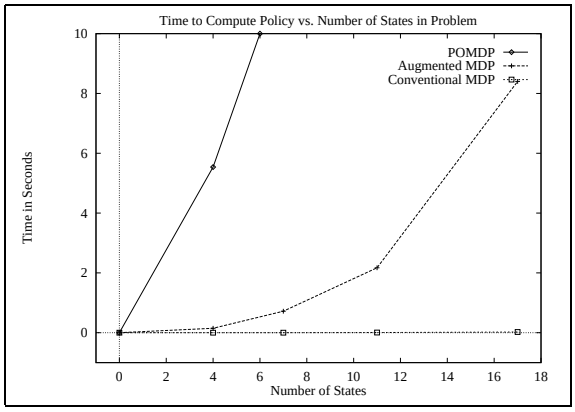


Figure 10: A comparison of the solution times for the full POMDP, augmented MDP and conventional MDP. The full POMDP is in fact exponential in growth as a function of the number of states.

Figure 10 demonstrates the need for a faster solution method for POMDP problems. The graph depicts number of states compared to solution time, for the Incremental Improvement full POMDP algorithm, the augmented MDP and the conventional MDP. The time to find an optimal POMDP policy is not only a function of the number of states but also the number of actions, observations and also the density of the transition and emission matrices, however this graph provides some rough intuition of the scalability of the three algorithms. The full POMDP topped out at 7 states, taking 729 seconds on a 400 MHz Pentium II. The augmented MDP appears to be quadratic in the number of states; however, the augmented MDP also has a strong dependence on the number of variables in the belief state representation and the discretisation of the belief state variables.

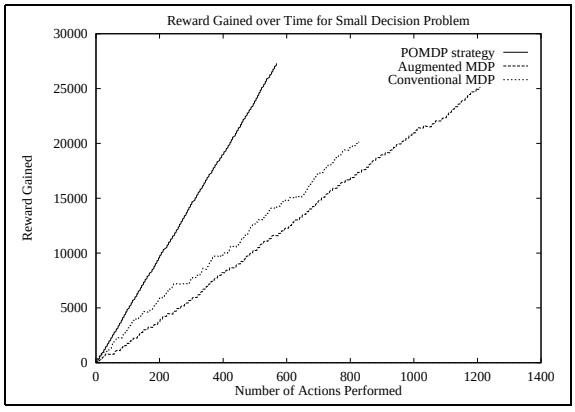


Figure 11: A comparison of the policies from the full POMDP solution, the augmented MDP and a conventional MDP solution, over a 7-state toy dialogue problem. The graph shows reward gained with each action. Note that while the full POMDP acquires the reward fastest, the Augment MDP policy still acquires reward faster than the conventional MDP.

Figure 11 shows the performance of the three algorithms, POMDP, augmented MDP and conventional MDP over

100 dialogues in the 7 state problem. The graph depicts total accumulated reward against number of actions, and the salient result is the slope of each line; the faster the policy accumulates reward, the better the policy. This graph demonstrates that the full POMDP accumulates the reward fastest. This is to be expected, because the POMDP policy is the optimal policy.

The augmented MDP outperforms the MDP, but does spend time asking confirmation questions in situations that the POMDP deems unnecessary. Furthermore, the augmented MDP, while outperforming the MDP solution does not do tremendously better. This is for two reasons; firstly, the problem domain is sufficiently small that the MDP does somewhat reasonably well; secondly, while the augmented MDP is not heavily penalised for asking confirmation questions, such actions do delay the accumulation of positive reward.

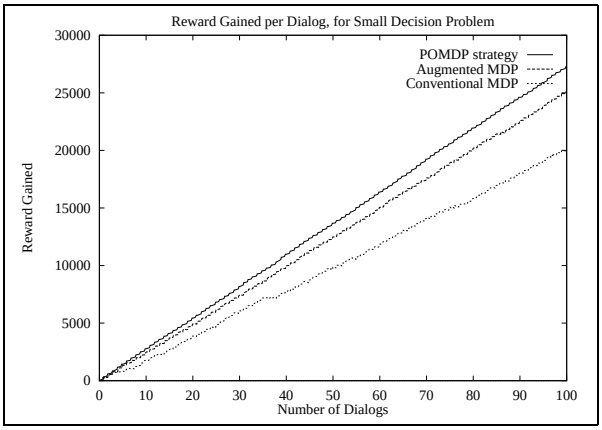


Figure 12: A comparison of the policies from the POMDP, augmented MDP and conventional MDP solutions over the 7-state toy problem. In this graph, the reward is per dialogue, to avoid penalising policies that for confirmatory actions that delay the eventual reward.

If we examine the accumulation of reward per dialogue as shown in figure 12 (as compared to reward per action as shown in figure 11), we see that the augmented MDP is closer to the optimal POMDP policy. The problem domain is still too constrained to demonstrate the failure of the MDP policy.

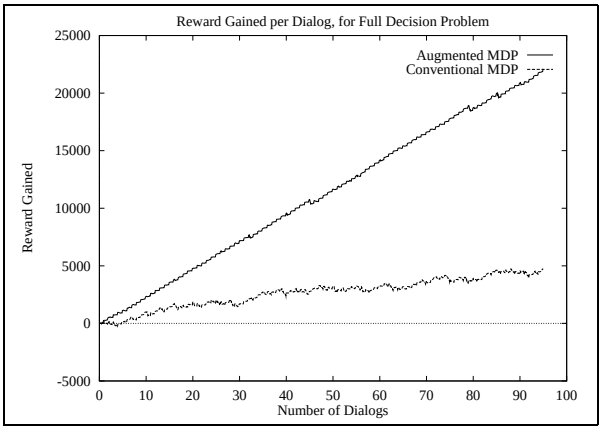


Figure 13: A comparison of the policies from the augmented MDP and conventional MDP solutions, over the full 17-state problem. Again, the graph depicts reward accumulated per dialogue, over the course of several dialogues.

Figure 13 shows the performance of the augmented MDP compared to the conventional MDP dialogue manager over the full 17 state problem. Because of the number of states, no POMDP solution could be computed for this problem. The augmented MDP clearly outperforms the conventional MDP strategy, as it more than triples the total accumulated reward over the lifetime of the strategies, although at the cost of taking longer to reach the goal state in

each dialogue. Table 4 breaks down the numbers in more detail. The average reward for the augmented MDP is 18.6 per action, which is the maximum reward for most actions, suggesting that the augmented MDP is taking the right action about 95% of the time. Furthermore, the average reward per dialogue for the augmented MDP is 230 compared to 49.7 for the conventional MDP, which suggests that the conventional MDP is making a large number of mistakes in each dialogue.

Augmented MDP	
Average Reward	18.6 +/- 57.1
Average Dialogue Reward	230.7 +/- 77.4
Conventional MDP	
Average Reward	3.8 +/- 67.2
Average Dialogue Reward	49.7 +/- 193.7

Table 4: A comparison of the rewards accumulated for the two algorithms using the full model in simulation.

Finally, the standard deviation for the augmented MDP is much narrower, suggesting that this algorithm is getting its rewards much more consistently than the conventional MDP.

4.4. Verification of Model on Users

We verify the utility of the POMDP by testing the model on actual human users. It must be emphasised that the user testing of the robot is still preliminary and therefore the experiment presented here cannot be considered a rigorous demonstration. However, table 5 shows some promising results.

The experiment consisted of having users interact with the mobile robot under a variety of conditions. The users tested both the augmented MDP, and an implementation of a conventional MDP dialogue manager. Both planners used exactly the same model. The users were presented first with one manager, and then the other, although they were not told which manager was first and the order varied from user to user randomly. The user labelled each action from the system as "Correct" (+100 reward), "OK" (-1 reward) and "Wrong" (-100 reward). The "OK" label was used for responses by the robot that were questions (i.e., did not satisfy the user request) but were relevant to the request, e.g., a confirmation of TV channel when a TV channel was requested.

		Augmented MDP	Conventional MDP
User 1	Reward Per Action	52.2	24.8
	Errors per request	0.1 +/- 0.09	0.55 +/- 0.44
	Time to fill request	1.9 +/- 0.47	2.0 +/- 1.51
User 2	Reward Per Action	36.95	6.19
	Errors per request	0.1 +/- 0.09	0.825 +/- 1.56
	Time to fill request	2.5 +/- 1.22	1.86 +/- 1.47
User 3	Reward Per Action	49.72	44.95
	Errors per request	0.18 +/- 0.15	0.36 +/- 0.37
	Time to fill request	1.63 +/- 1.15	1.42 +/- 0.63

Table 5: A comparison of the rewards accumulated for the two algorithms using the full model on real users, with results given as mean +/- std. dev.

For three different users, the system performed differently, compensating for the speech recognition accuracy which varied significantly between users. Notice that in user #2's case, the augmented MDP manager takes longer to satisfy the requests, but in general gains more reward per action. This is because the speech recognition system generally had lower word-accuracy for this user, either because the user had unusual speech patterns, or because the acoustic signal was corrupted by background noise.

By comparison, user #3's results show that in the limit of good sensing, the augmented MDP policy approaches the MDP policy. This user had a much higher recognition rate from the speech recogniser, and consequently both the augmented and conventional MDP acquire rewards at equivalent rates, and satisfied requests at similar rates.

5. Proposed Research

I have presented a method for finding approximate solutions to POMDPs, and applied the method to two different task domains on mobile robots. The augmented MDP avoids the short-sightedness of greedy heuristics while at the same time avoiding the computational complexity of exact solutions. However, a number of open questions remain. I propose 5 major directions of research using the augmented MDP:

- Theoretical analysis of Augmented MDP
- Developing better sufficient statistics
- Dynamic observation probabilities
- Speech Dialogue Management
- Mobile robot navigation and exploration

5.1. Extending the Algorithm

Solution Optimality Currently, it is difficult to compare an augmented MDP solution to one provided by solving the POMDP exactly. In most instances, this is because finding the exact POMDP is essentially impossible. However, in cases where an exact POMDP solution can be found, the augmented MDP and exact POMDP policy need to be compared. In the 7-state speech dialogue experiment described earlier, the augmented MDP policy is clearly suboptimal because it is not the same as the POMDP and acquires reward more slowly.

One interesting experiment is to determine whether or not the augmented MDP is simply too conservative in estimating the current state. This can be tested by making information-gathering cost-free (or at least reward-neutral); if such an augmented MDP acquires reward at the same rate as the regular POMDP, then it may be possible to show that the augmented MDP generates the optimal policy in the absence of penalties for information-gathering.

Solution Bounds So far, we have made no claims about the bounds on the quality of the solution. If the parameterisation of the belief space is in fact to sufficient statistics, then the policy should indeed be identical to the optimal POMDP policy. However, a proof of this should be developed. Furthermore, an analysis of when the statistics are indeed sufficient needs to be developed.

Finally, an analysis of the impact of the parameterisation on the solution quality is required. It may be (indeed, it is hoped) that in many problem domains, although a representation may not completely capture all aspects of all belief spaces, the resultant policy is optimal or possibly ϵ-optimal.

The Brown Problem Base Cassandra has established a database of various POMDP problems from the literature (`http://www.cs.brown.edu/research/ai/pomdp/examples/index.html`). In order to test the generality of the augmented MDP algorithm, we should test it on the set of problems in the database. The problems generally have a small state set (most are < 100 states), but it would be an interesting exercise to determine which problems the augmented MDP fails on. This particular test can be run reasonably quickly. It seems likely that in the current incarnation, the augmented MDP will do badly on a number of examples. However, with the extensions to the algorithm proposed below, it may be possible to increase the number of problems that the augmented MDP can solve.

Parameterisation and Discretisation of the Belief Space Throughout the results presented here, the belief space was parameterised by the conjunction of the maximum-likelihood state and the entropy of the belief state. This may have been a reasonable representation for the mobile robot problem navigating in the particular environment of the museum, however it is not likely that this representation generalises to all robot environments. Furthermore, this representation is completely unsupported for the speech domain. The results for the speech domain suggest that the entropy provides some information guiding the policy but it is likely that the policy is quite sub-optimal.

A closed form algorithm for computing the correct parameterisation to provide a set of jointly sufficient statistics for the belief state is unlikely in all domains. However, by generating example trajectories through the state space, it may be possible to learn appropriate parameterisations. Model selection is in general a difficult problem. However, domain-specific knowledge, such as a map in the case of mobile robot navigation, may also be useful for guiding the search for appropriate parameterisation.

The discretisation of the belief space is currently completely arbitrary. Example trajectories through the state space might also be useful for training the system on an appropriate discretisation level.

Dynamic Observation Probabilities The sensor model for the laser range sensor used by the mobile robot navigation problem is completely static. However, the speech model used by the dialogue management is not static. Sphinx provides probability estimates for the speech utterances at run-time. The emission probabilities used during planning are average estimates. Consequently, the policy chosen during execution may be optimal over several dialogues, but may not be optimal for a single dialogue. One approach may be to replan every dialogue after receiving every utterance using the latest observation model, or update the emission probabilities to a new average value and replan. The current speech domain is sufficiently low-complexity that the augmented MDP can be replanned in near real-time. However, as the domain increases in complexity it is not clear that re-planning is the correct strategy. Furthermore, it is not clear that re-planning is completely necessary. It may be that simply updating the belief appropriately with the emission probabilities given at runtime is the optimal policy; however, an analysis of the performance of the planner is needed under these conditions.

It is worth noting that this particular problem is not specific to speech recognition. A number of real world processes exhibit similar dynamic probabilities; for example, a diagnostic test may return a subset of possible hypotheses about medical illnesses, but it is impossible to know beforehand which hypotheses will be in the subset. Planning using the powerset of hypotheses is one alternative but is generally impractical.

5.2. Dialogue Management

Dialogue Management Experimental Results The dialogue management experiments consist solely of simulation at the moment. The first experiment that needs to be performed is a user study, to verify that the dialogue manager using the augmented MDP does indeed outperform the conventional MDP. Such an experiment would involve asking users to interact with the system, and allowing them to assign numerical values to the system's responses that correspond to both positive and negative rewards.

Preliminary user studies indicate that the augmented MDP will probably perform badly due to improper tuning of the model parameters. The model will almost certainly have to be revised several times before the power of the augmented MDP can be tested properly. It is worth noting that the field of dialogue management has produced different methods for evaluating user models (Aust and Ney, 1998; Litman and Pan, 1999). It may be worth implementing some of the different testing methodologies to determine if there are different regions where the augmented MDP performs better or worse.

Generating More Natural Dialogues We have not yet achieved the goal of natural dialogue with the robot. Dialogue with the robot is still characterised by an unnatural rigidity. This problem is at least in part caused by the limited set of domains and vocabulary, however, part of the problem is also the model.

In particular, there are two aspects of conversation that are poorly modelled by our system. The first aspect consists of speech acts that do not accomplish any particular task, but serve to indicate to the other party that the listener is still

engaged. Currently, the robot remains silent, and in general motionless, until the entire utterance is complete. This leads to slow interaction, such as can be found on a satellite telephone conversation with high lag-time.

Secondly, the model assumes that every interaction with the user follows some essentially linear trajectory until the user's task is satisfied. The transition probabilities in the current model do allow for the system to change tasks in mid-trajectory, however, such a change is unlikely, and is more representative of the user making a mistake than of a genuine change of interest.

Both of the problems alluded to above are a consequence of the model structure. In general, providing the correct model by hand for natural interaction is very difficult, and can only become harder as the models become more complex and represent richer interactions. The reward structure is very difficult to encode, and is the principle reason for both the assumption of an essentially linear trajectory from task onset to task completion, and the lack of speech acts that encourage interaction but do not achieve some easily identifiable goal.

However, machine learning has a number of techniques for learning such models. Van den Bosch in particular has examined a case-based learning technique in particular for learning speech acts that encourage interaction without achieving some measurable goal, learned from training data of two human users interacting (van den Bosch, 1999). Case-based learning would not be appropriate for learning for our probabilistic model, but using human-human interaction training data with a more appropriate learner is likely to improve the quality of the dialog manager substantially.

Learning Confirmation Actions A second area where machine learning could be of use is in learning the confirmation actions. Currently, the confirmation actions and the accompanying transition models are provided by hand. However, not all actions may always be needed. It would be preferable for the POMDP solver to reflect upon which states will possibly need to be disambiguated, and request the human modeller to provide the appropriate speech actions that will do so. For example, in the speech dialogue domains given earlier, it is unlikely that the system will ever need to distinguish between the states `want_nbc` and `want_tomorrow_weather`, and only those states. Therefore, there is no confirming action that corresponds to such a disambiguation. It would be extremely useful for the system to learn the distribution of uncertainty it experiences, and request specific confirming actions. One possibility is to learn these actions from human-human interaction, but this seems a much harder problem.

Integrating Multiple Observations Finally, the dialogue manager currently has no way of integrating non-speech cues as observations, in particular data from the touchscreen and other sensors. There are two known ways of integrating multiple observations from different sources. The first is two construct new, compound observations from the single observations, such as compounding a button press and a speech utterance into a single observation. However, this has the problem of generating exponentially many possible observations. Furthermore, for real-valued observations (such as from a camera or the laser-range finder), such an approach may not be feasible, although Simmons & Koenig used a similar approach by implementing a "virtual" sensor that represented features extracted from multiple observations (Simmons and Koenig, 1995). A second approach is to integrate the observations sequentially, separated by some null-action. However, this approach contains an independence assumption that is rarely valid, in particular for non-speech actions that are meant to disambiguate the accompanying speech actions. For this particular problem, there are two promising approaches; the first is to use natural language processing to integrate the non-speech observation with the speech observation into a single speech observation for situations where such integration is reasonable, otherwise discard one or the other observation. The second approach is to construct a virtual sensor that returns symbolic features that are constructed from multiple sensor streams including both speech and non-speech observations.

5.3. Mobile Robot Navigation

Mobile Robot Experimental Results The experimental results in both the robot navigation and speech dialogue management are both preliminary at best. In particular, the evaluation of the augmented MDP compared to existing algorithms needs to be further developed.

Specifically, in the robot navigation case, it is not sufficient show only that the certainty of the robot's position is increased. If the final position of the robot is no more accurate than for the conventional planner then the augmented

MDP may in fact be a worse planner, because the robot's belief in its position does not reflect reality. The conventional planner combined with the probabilistic localisation algorithm should at least reflect the reality that the robot is unsure of its position, even if it does nothing to remedy this problem.

One experiment that should be performed is to compare the ground-truth position of the robot with the maximum-likelihood position, and show that the maximum-likelihood position of the belief at the goal is on average statistically closer to both the true position and the goal when using the augmented MDP compared to a conventional planner. Another interesting experiment would be to examine the difference between the maximum-likelihood position and the true position over the entire length of the path; it seems likely, at least with the kinds of trajectories seen earlier, that the positional accuracy will be higher along the entire length of the path. Such a side-effect would be desirable, although the POMDP does not formally guarantee it.

Another experiment suggested by Wolfram Burgard would be to examine the positional accuracy as a function of robot speed. Given a fixed frequency of sensor data, travelling faster can be viewed as sensing less often. By examining the average positional accuracy (either along the trajectory or at the goal), it may be possible to pick an optimal and environment-dependent travel speed for the mobile robot.

Robot Exploration It may be possible to leverage the use of POMDPs to improve robot exploration, in the context of concurrent mapping and localisation (Thrun et al., 1998). As a robot explores, its belief state about its position in the world becomes increasingly uncertain. At some time, the robot must return to the known, already explored environment and update its position estimate with the existing map before integrating new map information.

Typically, planning for exploration and mapping involves local control rules; the decision whether to continue exploring or to return to already explored space and integrate the new spatial information into the existing map is usually made on the basis of local characteristics of the robot or the environment. Any kind of global planning is difficult to do in an unknown environment. However, there are two possible avenues for a non-heuristic POMDP approach to be applied to exploration, to generate more robust exploration trajectories.

The first is that the POMDP may allow for a principled ways of predicting whether or not the robot can return to known space with a known position. For example, if the known space contains a funnel, then the robot can explore much further and be able to integrate the new sensor data reliably than if the robot has to track its position along its entire trajectory. Local control rules will not be able to identify the existence of such a funnel or make use of it.

Secondly, the POMDP can be used for disambiguating different parts of the environment. A question that arises in exploration and mapping is one of recognising the same location twice in the environment, especially if the robot enters the same location twice but from different directions each time. Rehearsal procedures as described by Kuipers and Byun (1991) are often used for resolving such ambiguities. However, determining such a procedure is not trivial. The true identity of a location that is putatively the same as some other location in the environment can be modelled as an unobservable, or partially observable, feature of that location. A POMDP planner can then be used to generate the optimal path for detecting the identity of the new location.

6. Contributions

- I have proposed a novel algorithm for finding POMDP policies. I have developed this algorithm under a certain set of assumptions. I propose to make this algorithm more general by relaxing these assumptions.
- I have demonstrated the utility of POMDPs on two real world problems, robot navigation and speech dialogue management. POMDPs have been used for robot navigation previously, but my application is for a much larger state space and a metric, as opposed to topological representation.
- I have proposed a novel representation for natural language dialogue, and proposed that model parameters as well as confirmation actions can be learned for more natural dialogue management.
- I have proposed a method of using POMDP navigation for generating more robust exploration trajectories.

Future Schedule

Fall 2000

- Develop learning method for estimating number and type of jointly sufficient statistics over belief space and discretisation level
- Test learning method on larger dialogue problems with real users
- Develop closed-form modelling of belief-space compression

Spring-Summer 2001

- Develop handling of dynamic observation probabilities
- Develop theoretical analysis of augmented MDP
- Analyse compression on solution quality
- Perform better user testing of larger speech dialog models
- Develop model for integrating non-speech cues into dialog

Fall 2001

- Test full augmented MDP method on University of Brown POMDP database
- Finish mobile robot experiments
- Extend navigation to concurrent mapping and localisation

Spring-Summer 2002

- Continue extension of navigation to concurrent mapping and localisation
- Complete written thesis
- Defend

References

Aust, H. and Ney, H. (1998). Evaluating dialog systems used in the real world. In *Proc. IEEE ICASSP*, volume 2, pages 1053–1056.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, NJ.

Black, A., Taylor, P., and Caley, R. (1999). *The Festival Speech Synthesis System*, 1.4 edition.

Burgard, W., Fox, D., Hennig, D., and Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids. In *AAAI*.

Cassandra, A. "Tony's POMDP file repository". http://www.cs.brown.edu/research/ai/pomdp/examples/index.html.

Cassandra, A., Littman, M. L., and Zhang, N. L. (1997). Incremental pruning: A simple, fast, exact algorithm for partially observable markov decision processes. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)*.

Cassandra, A. R., Kaelbling, L., and Kurien, J. A. (1996). Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ Interational Conference on Intelligent Robotic Systems (IROS)*.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proc. 12th Nat. Conf. on Artificial Intelligence*, Seattle, WA.

Denecke, M. and Waibel, A. (1997). Dialogue strategies guiding users to their communicative goals. In *Proc. Eurospeech*.

Fox, D., Burgard, W., and Thrun, S. (1998). Active Markov localization for mobile robots. *Robotics and Autonomous Systems*.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Koenig, S. and Simmons, R. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement learning algorithms. *Machine Learning Journal*, 22:227–250.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using Markov decision process for learning dialogue strategies. In *Proc. ICASSP*.

Litman, D. J. and Pan, S. (1999). Empirically evaluating an adaptable spoken dialogue system. In *Proceedings of the 7th International Conference on User Modeling (UM)*, pages 55–64, Banff, Canada.

Littman, M., Cassandra, A., and Kaelbling, L. (1995). Learning policies for partially observable environments: scaling up. In *Machine Learning: Proceedings of the Twelyth International Conference*, pages 362–370, San Francisco, CA.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, Providence, RI.

Mahadevan, S. (1998). Partially observable sem-markov decision processes: Theory and application to engineering and cognitive sciences. In *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes*, Orlando, FL.

Monahan, G. E. (1982). A survey of partially observable Markov decision processes. *Management Science*, 28(1):1–16.

Niimi, Y. and Kobayashi, Y. (1996). Dialog control strategy based on the reliability of speech recognition. In *Proc. ICSLP*.

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

Parr, R. and Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the 14th International Joint Conferences on Artificial Intelligence*.

Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

Ravishankar, M. (1996). *Efficient Algorithms for Speech Recognition*. PhD thesis, Carnegie Mellon.

Roy, N., Burgard, W., Fox, D., and Thrun, S. (1999). Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proc. International Conf. Robotics and Automation*, Detroit, MI.

Roy, N. and Thrun, S. (1999). Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12, pages 1043–1049.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sim, R. and Dudek, G. (1998). Mobile robot localization from learned landmarks. In *Proceedings of the IEEE/RSJ Interational Conference on Intelligent Robotic Systems (IROS)*.

Simmons, R. and Koenig, S. (1995). Probabilistic navigation in partially observable enviroments. In *Proceedings of the 14th International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 1080–1087, Montreal, Canada.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999). Reinforcement learning for spoken dialog systems. In *NIPS*.

Smallwood, R. D. and Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088.

Sondik, E. (1971). *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, Stanford, California.

Takeda, H., Facchinetti, C., and Latombe, J.-C. (1994). Planning the motions of mobile robot in a sensory uncertainty field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(10).

Thrun, S. (1999). Monte carlo pomdps. In Solla, S. A., Leen, T. K., and Müller, K. R., editors, *Advances in Neural Processing Systems*, volume 12.

Thrun, S., Fox, D., and Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 431.

Torrance, M. C. (1994). Natural communication with robots. Master's thesis, MIT Dept of E.E.and C.S.

van den Bosch, A. (1999). Instance-family abstraction in memory-based language learning. In Bratko, I. and Dzeroski, S., editors, *Machine Learning: Proceedings of the Sixteenth International Conference (ICML)*, pages 39–48, Bled, Slovenia.

Westphal, M. and Waibel, A. (1999). Towards spontaneous speech recognition for on-board car navigation and information systems. In *Proc. Eurospeech*, Budapest, Hungary.

Young, S. (1990). Use of dialogue, pragmatics and semantics to enhance speech recognition. *Speech Communication*, 9(5-6).