A formal framework for robot learning and control under model uncertainty

Robin JAULMES, Joelle PINEAU and Doina PRECUP
School of Computer Science
McGill University
Montreal, QC CANADA H3A 2A7

Abstract— **While the Partially Observable Markov Decision Process (POMDP) provides a formal framework for the problem of robot control under uncertainty, it typically assumes a known and stationary model of the environment. In this paper, we study the problem of finding an optimal policy for controlling a robot in a partially observable domain, where the model is not perfectly known, and may change over time. We present an algorithm called MEDUSA which incrementally learns a POMDP model using queries, while still optimizing a reward function. We demonstrate effectiveness of the approach for a simple scenario, where a robot seeking a person has minimal a priori knowledge of its own sensor model, as well as where the person is located.**

I. Introduction

Most approaches for robot planning and control assume there is a known model that quantitatively describes the robot's dynamics and sensors, as well as other agents and the environment. This is convenient when available, and solutions have been produced for large-scale high-dimensional domains using a wide variety of algorithms. However these methods are severely limited in applications where writing down the exact model is time-consuming and error-prone, or when the model changes over time. A number of ad-hoc approaches are available for dealing with this, often involving re-planning. However the model uncertainty is not generally considered as an integral part of the initial planning and control algorithm.

This paper proposes a formal framework for robot planning and control under model uncertainty. The framework accommodates uncertainty in the dynamics (the robot's own, as well as the environment's and other agents'), uncertainty in the sensors, and uncertainty in the very model that describes these dynamics and sensors. The framework makes several assumptions: it assumes the model can be described probabilistically, it assumes the robot can interact with the environment to learn the model, and finally it assumes access to an operator (or highly precise sensor) who can answer queries about the state of the system. The queries are automatically selected as part of the planning exercise so as to maximize information gain. The approach we describe suggests a new paradigm for shared human-robot learning in unknown environments.

Our approach is based on the Partially Observable Markov Decision Process (POMDP) [7]. POMDPs provide a sound decision-theoretic framework for planning under uncertainty. Due to recent advances in algorithmic techniques, POMDPs have been applied in large domains where the state of the robot and other agents in the environment is only partially observable [10]. These **model-based** techniques however require an exact and stable model of the robot's sensors, dynamics and environment. A number of **experience-based** approaches have also been proposed for the POMDP, which do not assume a known model, but rather learn it from direct experience with the environment [2], [12], [1]. However these typically require excessive amounts of data. We would prefer using techniques which offer a better compromise between model-based and experience-based.

The approach we describe, called MEDUSA, combines a partial model of the environment with direct experimentation, in order to produce solutions that are robust to model uncertainty, while scaling to larger robotics domains than experience-based approaches. Furthermore, because models may change over time (e.g. non-stationary environments), our approach is able to automatically track these changes and adapt the model accordingly. MEDUSA is inspired from the fields of active learning [3] and Bayesian reinforcement learning [4]. The core idea is simple: represent any prior knowledge about the robot and its environment in a set of Dirichlet distributions and update the parameters of the Dirichlets whenever the robot interacts with its environment or queries an oracle (e.g. operator, high-precision sensor) . To obtain a plan, we sample POMDP models from the Dirichlet distributions, and solve these using model-based techniques.

The original MEDUSA algorithm was presented in [5]. We discussed in [6] theoretical conditions under which MEDUSA converges to the correct model. We now present a novel and extended version of MEDUSA that includes learning from both queries and non-queries, as well as adaptation to non-stationary environments. We also provide the first empirical validation of MEDUSA for a robotic task. The main contribution of this paper is to present an improved framework for simultaneous learning and control that is appropriate for complex robot systems.

II. Robot platform

Many people who suffer from chronic mobility impairments, such as spinal cord injuries or multiple sclerosis, use a powered wheelchair to move around their environment. However factors such as fatigue, degeneration of their condition, and sensory impairments, often limit their ability to use standard electric wheelchairs. SmartWheeler (Figure 1) has been developed as a multi-functional intelligent wheelchair to assist individuals with mobility impairments in their daily locomotion, while minimizing physical and cognitive loads. It is built on top of a commercially available Sunrise Quickie Freestyle, to which we have added laser range-finders, wheel odometers, a touch-sensitive graphical display, a voice interface, and an onboard computer.

The robot's primary task involves navigating in an indoor environment following high-level commands from the user (of the type "Go to the dinning room", "Find a caregiver", etc.) All mapping and navigation functions are provided by the Carmen robot navigation toolkit [9]. The learning and planning engine is provided by the MEDUSA system described in the current paper.

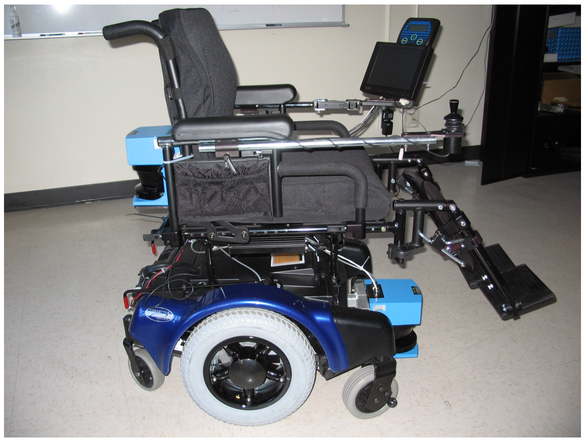


Fig. 1. SmartWheeler robot platform.

III. Partially Observable Markov Decision Processes

The MEDUSA learning and planning engine is based on the POMDP paradigm. This section reviews the basic concepts in POMDPs. We assume the standard POMDP formulation [7]. A discrete set of **states**, S, describes the task domain. A discrete set of **actions**, A, describes what the robot can do. A discrete set of **observations**, Z, describes what the robot can perceive.

Unlike other planning representations (e.g. STRIPS, MDPs), the assumption in POMDPs is that the state of the system is not known, but can only be perceived (partially) through the observations. The challenge is for the robot to pick a good sequence of actions, despite this state uncertainty. Formally stated: at each time step t, the agent is in an unknown state $s_t \in S$, it executes an action $a_t \in A$, arrives in an unknown state $s_{t+1} \in S$ and gets an observation $z_{t+1} \in Z$.

The dynamics of the POMDP are defined via a probabilistic representation of the task and environment as follows:

- state-to-state transition probabilities
  $T^a_{s,s'} = Pr(s_{t+1} = s' | s_t = s, s_t = a), \forall s \in S, \forall a \in A, \forall s' \in S$,
- observation emission probabilities
  $O^a_{s,z} = Pr(z_t = z | s_t = s, a_{t-1} = a), \forall z \in Z, \forall s \in S, \forall a \in A$.

It also assumes a deterministic reward function $R : S \times A \times S \times Z \rightarrow \Re$, where $R(s_t, a_t, s_{t+1}, z_{t+1})$ is a quantitative assessment of the usefulness of the corresponding experience-tuple $< s_t, a_t, s_{t+1}, z_{t+1} >$. Finally, there is a known discount factor $\gamma \in [0; 1]$ which quantifies preference for reaching the goal at the current time step, versus the next time step.

Since the state is not fully observable, robots operating under the POMDP framework keep track of a *belief state*, $b \in \Re^{|S|}$, which is a probability distribution over all states given the history of actions/observations experienced so far. The initial belief b_0 is usually specified by the designer. Subsequent beliefs b_t can be calculated by Bayesian updating:

$$b_t(s') = \frac{O^a_{s',z} \sum_{s \in S} T^a_{s,s'} b_{t-1}(s)}{\sum_{\sigma \in S} O^a_{\sigma,z} \sum_{s \in S} T^a_{s,\sigma} b_{t-1}(s)} \tag{1}$$

based on the most recent action, observation pair.

A policy is a function that associates an action to each possible belief state: $\pi(b) \rightarrow a$. Solving a POMDP means finding the policy that maximizes the expected discounted return:

$$E[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1}, z_{t+1}) | b_0]. \tag{2}$$

While finding an exact POMDP solution is often computationally intractable, due to the large number of possible belief states, many efficient methods have been proposed recently for finding approximate solutions. The results presented in this paper use the PBVI algorithm [10], however, the techniques MEDUSA is equally suited to other approximations.

While these approximate methods have been crucial in scaling POMDP solving to larger domains and robotic applications [11], [13], they require knowing the exact probabilistic distributions underlying the POMDP model. This is a challenge in many realistic robotic tasks. For example, in SmartWheeler, this would mean having a precise parametric description of: the robot's motion model, sensor noise models for the odometry, lasers and speech interface, and a probabilistic description of the human user's state transitions. Clearly, some components are easier to describe probabilistically than others, for example the motion model and laser noise model have been quantified previously in the literature. However the odometry noise for this robot is unknown, and user's state transition and speech patterns are difficult to quantify without data. MEDUSA provides a formal framework for combining a priori knowledge and experimental data in a Bayesian manner, to acquire high-quality model and policy for the task at hand.

IV. The MEDUSA algorithm

The core idea of MEDUSA is to represent the model uncertainty with a Dirichlet distribution over possible models, and to update directly the parameters of this distribution as new experience is acquired. We assume throughout that the reward function is fully known; this is in contrast to the standard reinforcement learning formulation [14]. We focus instead on learning $\{T^a_{s,s'}\}$ and $\{O^a_{s,z}\}$ which can be hard to specify correctly by hand.

This approach scales nicely: we need one Dirichlet parameter for each uncertain POMDP parameter, but the size of the underlying POMDP representation remains unchanged, which means that the complexity of the planning problem does not increase. However this approach requires the agent to repeatedly sample POMDPs from the Dirichlet distribution and solve the sampled models, to best select the next query.

A. Dirichlet Distributions

Consider a N-dimensional multinomial distribution with parameters $(\theta_1, \ldots \theta_N)$. A Dirichlet distribution is a probabilistic distribution over these parameters. The Dirichlet itself is parameterized by hyper-parameters $\alpha = \{\alpha_1, \ldots \alpha_N\}$. The likelihood of the multinomial parameters is defined by:

$$p(\theta_1 \ldots \theta_N | \alpha) = \frac{\prod_{i=1}^{N} \theta_i^{\alpha_i - 1}}{Z(\alpha)}, \text{ where } Z(\alpha) = \frac{\prod_{i=1}^{N} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i)},$$

and $\Gamma()$ is the standard gamma distribution. The maximum likelihood multinomial parameters $\theta_1^* \ldots \theta_N^*$ can be easily computed: $\theta_i^* = \frac{\alpha_i}{\sum_{k=1}^{N} \alpha_k}, \forall i = 1, \ldots N$. The Dirichlet distribution is convenient because its hyper-parameters can be updated directly from data. For example, if instance i is encountered, α_i should be increased (typically by 1). Also, we can sample from a Dirichlet distribution conveniently using Gamma distributions.

In the context of POMDPs, model parameters are typically specified according to multinomial distributions, thus our use of the Dirichlet distribution to capture model uncertainty. A separate Dirichlet distribution is specified for each state-action pair where either the transition probabilities or the observation probabilities are uncertain.

B. MEDUSA

The name MEDUSA comes from *Markovian Exploration with Decision based on the Use of Sampled models Algorithm*. It is an integrated learning and planning approach for sequential decision-making. We first give an overview of the basic algorithm, then discuss enhanced components designed to handle learning without an oracle, and learning in non-stationary environments.

The algorithm is as follows. First, we assume that initial Dirichlet parameters are given, representing both *a priori* knowledge of the model, and uncertainty over model parameters. This is the place to encode any prior knowledge about the robot's motion or sensor models.

Next, we sample a number of POMDP models according to this prior. Each model is a candidate representation of the robot's properties and task domain. In practice, a small number of models is usually sufficient to obtain good results while limiting computation (e.g. we used 20 for the experiments below.)

MEDUSA then computes an (approximately) optimal policy for each of the sampled models. The resulting policy describes optimal behavior for the robot under the assumed model parameters. MEDUSA also computes the weight (likelihood) of the model under the Dirichlet distributions.

Next, at each time step, one of the models is chosen at random (with probability equal to the weight of that model) and the corresponding optimal action is applied. This allows us to obtain reasonable execution performance throughout the learning process, while focusing learning in regions of the state space most often visited by good policies.[1]

Each time an action is taken, it is followed by an observation. The agent can then decide whether to query the oracle for the true identity of the hidden state. A discussion of the criteria used to decide whether or not we do a query is differed to the next section. If we do a query, the Dirichlet distributions are updated according to the outcome of that query. If we choose not to do a query, the Dirichlet parameters can also be updated; this is described in the next section as well. We then repeat these steps, starting with the selection of the next action. As time goes and the model improves, the system should choose to do fewer and fewer queries. In [6], we study the theoretical properties of the basic MEDUSA algorithm, and show that under certain conditions (in the limit of infinite exploration, and infinite number of sampled models), the algorithm is guaranteed to converge to the correct model.

Many practical considerations arise in the implementation of this basic algorithm. We discusse a few of these that are most relevant to robotic applications. First, note that whenever Dirichlet parameters are updated, the weight of the sampled models changes. At regular intervals, MEDUSA drops models whose weight falls below a threshold, and re-samples new models.

The core assumption in MEDUSA is the availability of the oracle to provide the robot with exact information about the current (and next) state. In many tasks it is possible (but very costly) to have access to the full state information; it can require asking a human to label the state, or using a high-precision sensor to recover the information. In practice, this assumption can be relaxed significantly. For example, while a robot experiments with a new environment, it can query the oracle to obtain exact state information *only when this is deemed necessary* since the state information is used only to improve the model, but not in the action selection process. This is discussed in the next section. For the same reason, there can be delays between the query request and the query processing (as long as we do not redraw models after every step). This is useful for realistic system deployment where a robot needs to carry out actions at a faster rate than the operator can be consulted. The answer to the query can also be noisy. Finally, learning can be performed strictly from experience (without resorting to the oracle), in which case our method is equivalent to strict experience-based methods.

Those familiar with POMDP representations of robotic domains will quickly notice that model parameters are often correlated. For example, the sensor noise may be identical regardless of the robot's position. And the noise induced by the motion is equivalent in all areas where the surface is the same. In effect, there are a set of *hyper-parameters* that describe the dynamics of the domain. Standard reinforcement learning makes no allowance for learning such hyper-parameters; the properties of each state-action combination

[1]For those familiar with reinforcement learning, this is our *exploration policy*. Other exploration policies could be used, (e.g. ϵ-greedy [14]), however we obtain good empirical results with this.

must be learned separately. MEDUSA on the other hand is perfectly suited to handle such situations. The key is to specify the hyper-parameter structure a priori by simply definition a function that maps any possible transition or observation parameter to a specified hyper-parameter. The hyper-parameters are updated directly whenever queries are made and answered. By varying the number of hyper-parameters, we are able to trade-off the number of queries versus model accuracy and performance. Also, any transition or observation parameter that is known with full certainty can be fixed a priori, rather than learned through queries. These modifications can be extremely useful in constraining the learning problem. The result is a highly expressive framework to represent, and reason about, model uncertainty.

C. Non-Query Learning

As suggested above, it is not necessary to make a query at every step, and furthermore it is possible to update the model based solely on evidence, using *non-query learning*. This requires (1) deciding when to query and (2) if we do not query (or if the query isn't answered), then using the information we have from the action-observation sequence and knowledge extracted from the previous queries to update the Dirichlet parameters. We begin with this second point.

To do an efficient non-query learning we introduce the concept of an **alternate belief** β. For each model $i = 1, \dots, n$, we keep track of such an alternate belief, (denoted β_i) in addition to the standard belief (denoted b_i). The alternate belief is updated in the same way as the standard one (see Eqn 1) with the difference that whenever a query is answered, β_i is updated to reflect the exact state information, whereas b_i tracks belief uncertainty according to model i's parameters (ignoring the state information). The alternate belief allows us to keep track of the information available from the latest query, when applying non-query learning. For notational convenience, we also define:

$$\beta_i(s,s') = \frac{[O_i]^a_{s',z}\,[T_i]^a_{s,s'}\,\beta_i(s)}{\sum_{\sigma\in S}[O_i]^a_{\sigma,z}\,[T_i]^a_{s,\sigma}\beta_i(s)},$$

and:

$$\beta'_i(s') = \sum_{s\in S}\beta_i(s,s').$$

Now assume the system decides not to make a query and sees the standard experience-tuple: $< a,z >$. The *non-query learning* updates all possible state transition hyper-parameters according to Equation 3 and all possible observation hyper-parameters according to Equation 4.

$$\alpha(s,a,s') \leftarrow \alpha(s,a,s') + \lambda\sum_{i=1}^{n} w_i\beta_i(s,s') \quad (3)$$

$$\alpha(s',a,z) \leftarrow \alpha(s',a,z) + \lambda\sum_{i=1}^{n} w_i\beta'_i(s') \quad (4)$$

Next, consider the decision of when to make a query and when to pass. In [5] we discussed a decision-theoretic optimal approach for deciding this. The method presented there was highly impractical for all but the smallest problems (e.g 2 states, 2 actions). We therefore consider the use of different heuristic indicators that are better suited to large-scale systems:

- `Variance`$= \sum_i w_i(V^{\pi_i} - \hat{V})^2$
  measures the variance over values computed by each model, V^{π_i}. Experiments show that this indicator captures efficiently how much learning remains to be done.
- `InfoGain`$=$
  $\sum_{i=1}^{n}[w_i\sum_{s,s'\in S^2}[\beta_i(s,s')(\frac{1}{\sum_{\sigma\in S}\alpha(s,a,\sigma)} + \frac{1}{\sum_{z\in Z}\alpha(s',a,z)})]]$
  measures the quantity of information that a query could bring. Note that we assume $\alpha(s,a,\sigma) \to \infty$ (and similarly for $\alpha(s',a,z)$) when the corresponding parameter is fully known a priori. This indicator is useful to reject queries in cases where the model is already sufficiently certain.
- `AltStateEntropy`$=$
  $\sum_{s\in S} -[\sum_{i=1}^{n}\beta_i(s)]\log(\sum_{i=1}^{n}\beta_i(s))$
  measures the entropy in the mean alternate belief. It is a good indicator of how much knowledge has been lost since the last query. When the state is still well identified, non-query learning is appropriate, but a query could be useful if the alternate belief has high entropy.
- `NumberOfQueries` counts the number of queries already answered.

We combine these heuristics to decide when to perform *query* versus *non-query* learning, using a simple decision-rule. We do a query iff:

$$(\text{AltStateEntropy} > \varepsilon_1) \wedge (\text{InfoGain} > \varepsilon_2) \wedge \quad (5)$$
$$((\text{Variance} > \varepsilon_3) \vee (\text{NumberOfQueries} < \text{Nmin}))$$

The first condition ensures that no query should be done if enough information about the state remains from previous queries. The second condition ensures that a query will not be made if it does not bring direct information about the model, which is the case if we are in a subpart of the model we know very well. The third condition ensures that we will stop doing queries when our model uncertainty does not have any influence on the expected return. Note for this term that considering the number of queries already done is necessary in some cases (especially in a learning setup with a completely uninformed prior) since at the beginning all the models can be equally bad, which can give a very low value for the Variance heuristic.[2]

D. Handling non-stationarity

There are many interesting robotic domains where the model parameters change over time. For example, slow decay in the wheel alignment may change the robot motion model, or a person interacting with the robot may change preferences over time. We would like our learning algorithm to be able to handle such non-stationarity in the model parameters. This is possible in some standard planning and control algorithms, but only through re-planning, which of course requires deciding when to initiate replanning. In MEDUSA,

[2]Other heuristics were considered—e.g. `PolicyEntropy` (entropy of the policy over models), `BeliefVariance` (variance on the belief state)—but were rejected due to poor empirical performance.

```
1) Define $\lambda \in (0,1)$, the learning rate.
2) Initialize Dirichlet distributions $\alpha(s,a,s')$, $\alpha(s',a,z)$, for any uncertain transition and observation distributions.
3) Sample $n$ models $P_1, \dots P_n$ from these distributions.
4) Compute initial model likelihoods: $p_i^0,\ i = 1, \dots n$.
5) Solve each model: $P_i \rightarrow \pi_i,\ i = 1, \dots n$.
6) Initialize the belief for each model $b_i = b_0,\ i = 1, \dots n$.
7) Initialize the *alternate* belief $\beta_i = b_0,\ i = 1, \dots n$.
8) Initialize the history $h = \{\}$.
9) Repeat:
   a) Compute the optimal action: $a_i = \pi_i(b_i),\ i = 1, \dots n$.
   b) Randomly pick (and execute) a policy action according to model weights:
      $\pi_i(b_i)$ is chosen with probability $w_i = \frac{p_i}{p_i^0}$, where $p_i^0$ is the *original* likelihood of model $i$ and $p_i$ is the *current* likelihood of model $i$.
   c) Acquire observation $z$.
   d) Update history trace $h = \{h, a, z\}$
   e) Update $b_i$ and $\beta_i,\ i = 1, \dots n$.
   f) Determine if any learning should be done by checking Eqn 5
      • If yes, observe query outcome $(s, s')$, and update Dirichlet parameters accordingly:
        $\alpha(s,a,s') \leftarrow \alpha(s,a,s') + \lambda$
        $\alpha(s',a,z) \leftarrow \alpha(s',a,z) + \lambda$
      • If no, update Dirichlet parameters using non-query learning according to Equations 3 and 4.
   g) Re-compute the model likelihoods $p_i,\ i = 1, \dots n$.
   h) If model is non-stationary:
      $\alpha(s,a,s') \leftarrow \alpha(s,a,s') - \nu$
      $\alpha(s',a,z) \leftarrow \alpha(s',a,z) - \nu$
   i) At regular intervals:
      • Remove the model $P_i$ with the lowest likelihood
      • Draw new model $P_i$ according to current Dirichlet distributions
      • Solve $P_i \rightarrow \pi_i$
      • Update its belief $b_i = b_0^h$, where $b_0^h$ is the belief resulting when staring in $b_0$ and seeing history $h$
      • Update alternate belief $\beta_i$ similarly, but taking into account last query result.
```

TABLE I

THE MEDUSA ALGORITHM

non-stationarity can be handled by simply allowing recent experiences to be weighed more heavily than older experiences.

To allow for non-stationarity, each time we update one of the hyper-parameters, we multiply all the hyper-parameters corresponding to the associated multinomial distribution by a model decay factor: $\nu \in [0,1]$. This can be thought of as a decay weight over model certainty. It does not change the *most likely estimate* of any of the updated parameters, but it does diminish the *confidence* over the parameters. Note that the equilibrium confidence is defined as: $C_{max} = \lambda \frac{1}{1-\nu}$, which is attained after an infinite number of samples, and is an indicator of our confidence in past experience. This is high when we believe the model is stable.

This concludes our description of the MEDUSA algorithm. Table I provides a detailed description of MEDUSA, including implementation details. We now proceed with a discussion of preliminary experimental validation.

V. EXPERIMENTAL RESULTS

We now turn our attention to the application of MEDUSA in a hypothetical robot task domain. To begin our investigation, we integrated MEDUSA with the Carmen robot toolkit [9]. The Carmen toolbox has been widely used in the robotics community for the control of indoor mobile robots. It is particularly useful for validation of new algorithms because its simulator is known to be highly reliable and policies with good simulation performance can typically be ported without modification to the corresponding robot platform.

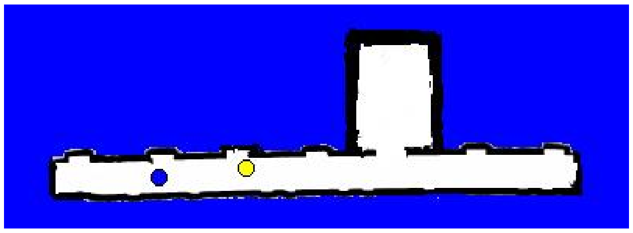


Fig. 2. Map of the environment used for the robot simulation experiment.

Our experiments thus far have focused on a scenario where the SmartWheeler must navigate in an environment, with the goal of autonomously finding a caregiver that is also mobile. We consider the environment shown in Figure 2. Planning and learning are done over a discretized version; the associated POMDP has 362 states, 24 observations and 5 actions. Execution assumes the continuous state representation and in that case belief tracking is done onboard Carmen using the full particle filter. Similar versions of this problem have been studied before in the POMDP literature under various names (Hide, Tag, Find-the-patient). Previous work always assumed a fully modeled version of this problem, where the person's location is unknown, but the person's motion model is precisely modeled, as are the robot's sensor and motion models.

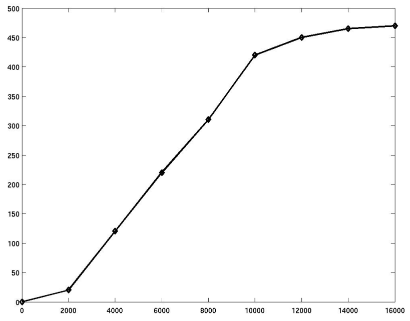
We now consider the case where in addition to not knowing the person's position, we are also uncertain about the person's motion model and the robot's sensor model. To avoid learning 362*362*5 transition and 362*5*24 observation parameters, we take advantage of symmetry in the environment to specify a smaller number of hyper-parameters. In total, MEDUSA is trying to learn 52 distinct parameters.

During learning, MEDUSA makes several queries about the state. Since there is no model uncertainty about the robot's motion, this is equivalent to asking the caregiver to reveal his/her position so that MEDUSA can infer his/her motion model. The answer to the queries can be provided by a human operator, though for convenience of carrying out multiple evaluations, in our experiments they are produced using a generative model of the caregiver.

As we can see from Figure 3, MEDUSA converges within roughly 12,000 time steps, after having received answers to approximately 9,000 queries. While this may seem large, it is orders of magnitude faster than experience-based approaches, which can require millions of steps to learn problems with less than a dozen states. We note however that experience-based approaches do not require an oracle. It is worthwhile pointing out that MEDUSA's oracle can in fact take the form of a high-precision sensor. It is realistic to assume for example that the caregiver will carry around a GPS sensor that can answer queries automatically during the learning phase, and that this will play the role of the oracle. In such a setup, 9,000 queries seems a small price to pay to obtain a full probabilistic model of the person's motion model.

We emphasize that the high number of queries required by MEDUSA for this problem is in large part due to the fact that the experiment assumed completely uninformed initial priors on the robot's sensor model and the caregiver's motion model (i.e. $\alpha(\cdot) = 0, \forall s, a, s', z$ in Step 2 of Table 1). Using a more informed prior would lead to faster learning, but would require more knowledge engineering. To reduce the number of queries, we could also build a simpler model with fewer Dirichlet parameters, in effect assuming stronger correlations between model parameters.

The main purpose of the results presented here is simply to establish the feasibility of learning probabilistic models for a realistic robotic domains with hundreds of states. Future investigations will be carried out to verify deployment of MEDUSA in a range of test domains and under various parameter configurations.


(a) Discounted return as a function of the number of time steps.

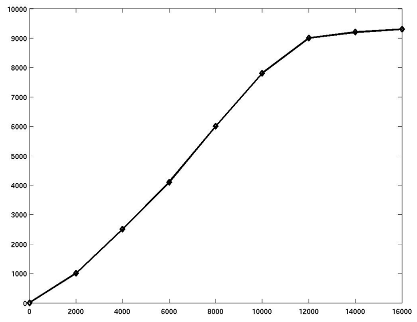
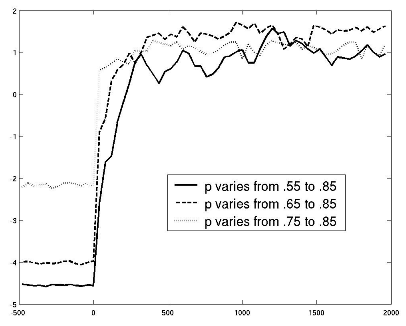

(b) Number of queries as a function of the number of time steps.

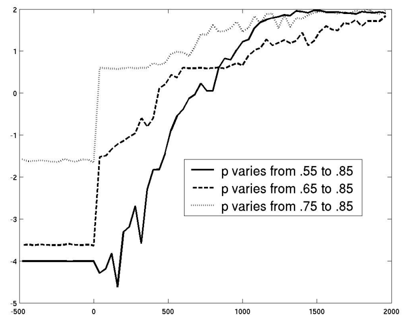
Fig. 3. Results for the robotic task domain.

Next, we consider the question of whether MEDUSA is robust to sudden changes in model parameters. This arises in non-stationary environments, where parameters can either (1) slowly drift over time (e.g. slow decay in the wheel alignments), or (2) change abruptly (e.g. person has an injury, which suddenly slows down his/her walking speed). Throughout these cases, MEDUSA should adapt to changes and learn the new correct model with high confidence. If the change in parameters is small, then non-query learning is sufficient, however if there are large changes, it is necessary to resort to queries.

We have not yet investigated MEDUSA's ability to handle non-stationary environments in the robot scenario above, however we consider a simple POMDP domain [7], where we assume that the probability of correctly detecting the target suddenly changes. Figure 4 summarizes the results. As expected, the speed at which MEDUSA learns the new model depends on the confidence (sum of Dirichlet parameters) prior to the change in parameter. When the confidence is low ($<$100), meaning that a weak model had been learned, the agent quickly adapts to the new parameters, even when there is a large shift in the model. However when confidence is high ($>$1000), then the agent takes many more steps to learn new parameter values. This confirms our hypothesis that the MEDUSA framework is able to handle non-stationarity in parameter values without the need for re-planning, or for explicitly detecting the change in parameters. This happens as an integral part of the learning process.



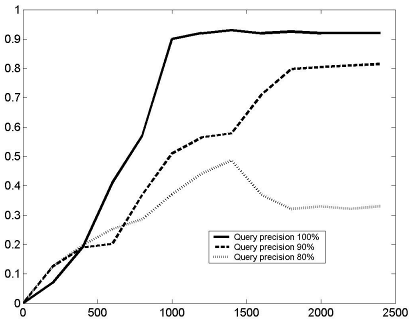
(a) The equilibrium confidence before the change is low (100).



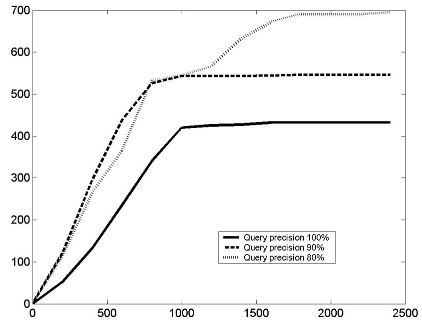
(b) The equilibrium confidence before the change is high (1000).

Fig. 4. Evolution of the discounted return as a function of the number of time steps. There is a sudden change in the parameter p at time 0.

Finally, we investigate MEDUSA's robustness to errors in the query responses. MEDUSA's query learning assumes that an oracle can provide exact state identification on demand. However it is more realistic to assume some amount of noise in the state identification provided by the oracle. This allows us to broaden the class of information sources we can use as an oracle. We do not yet have results for robustness to query noise in the robotic scenario above. However we have investigated the issue in the context of a simpler robot navigation domain [8]. We consider two cases. In the first, whenever a query is made, there is a 10% probability that the oracle will identify the wrong state (picking one at random), and a 90% that the oracle will give the correct answer. In the second case, the probability of a correct answer is 80%. As we see from Figure 5, the quality of the learned model is quite good when the oracle is correct in 90% of queries, though it takes more steps of query learning (and thus fewer steps of non-query learning) than when the oracle is always correct. The performance is significantly degraded for the higher error rate. However the 80% query precision model may not have converged yet, as we see in Figure 5b that the number of queries is still climbing.



(a) Discounted return as a function of the number of time steps.



(b) Number of queries as a function of the number of time steps.

Fig. 5. Results for queries with a noisy oracle.

VI. Discussion

Most approaches for robot planning and control assume access to a full model of the robot's dynamics, sensors and task domain. Such models are extremely useful when available, but are generally difficult and expensive to acquire, and become obsolete following changes in the robot or environment. This paper proposes a formal framework for robot planning and control in which reasoning about model uncertainty is an integral part of the approach.

The work we describe bears some resemblance to earlier techniques for decision-making in model-free POMDPs, which can learn and plan despite uncertainty in both state and model [2], [12], [1]. However these approaches have been scarcely used in robotics due to their intensive data requirements. MEDUSA offers a significantly more flexible trade-off between knowledge engineering (in the form of model priors) and data requirements (in the form of query answers and non-query experiences). This makes it particularly attractive for real-world domains where parts of the problem can be specified differently.

We demonstrate that the approach is applicable to learning non-trivial robotic tasks with significant model uncertainty. We also demonstrate applicability to non-stationary domains. The core assumption of MEDUSA is the availability of an oracle to provide state information. While this is a strong assumption, the framework and empirical results indicate that it can be relaxed in a number of ways. First, the oracle can be replaced by a high-quality sensor. Second, there is some tolerance for noise in the oracle's answers. And third, there can be a delay between the time the query is posed and when the answer is received. In the future, we will perform a thorough investigation of MEDUSA's robustness to a wide range of such experimental conditions. We will also investigate improved decision-theoretic methods for selecting when to do queries.

Acknowledgments

This work was made possible by funding from the Natural Sciences and Engineering Research Council Canada.

References

[1] Brafman, R. I. and Shani, G. “Resolving perceptual aliasing with noisy sensors” NIPS, 2005.
[2] Chrisman, L. “Reinforcement learning with perceptual aliasing: The perceptual distinctions approach” AAAI, 1992.
[3] Cohn, D. A., Ghahramani, Z. and Jordan, M. I. “Active Learning with Statistical Models” NIPS, 1996.
[4] Dearden, R.,Friedman, N.,Andre, N., “Model Based Bayesian Exploration” UAI, 1999.
[5] Jaulmes, R.,Pineau, J.,Precup, D., “Active Learning in Partially Observable Markov Decision Processes” ECML, 2005.
[6] Jaulmes, R.,Pineau, J.,Precup, D., “Learning in non-stationary Partially Observable Markov Decision Processes”. ECML Workshop on Reinforcement Learning in non-stationary environments, 2005
[7] Kaelbling, L., Littman, M. and Cassandra, A. “Planning and Acting in Partially Observable Stochastic Domains” Artificial Intelligence. vol.101, 1998.
[8] Littman, M., Cassandra, A. and Kaelbling, L. “Learning policies for partially obsevable environments: scaling up”. ICML, 1995.
[9] Montemerlo, M., Roy N. and Thrun, S. “Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit”. IROS, 2003.
[10] Pineau, J., Gordon, G. and Thrun, S. “Point-based value iteration: An anytime algorithm for POMDPs”. IJCAI, 2003.
[11] Pineau, J., and Gordon, G. ”POMDP Planning for Robust Robot Control”. ISRR, 2005.
[12] Shatkay, H., Kaelbling, L. “Learning topological maps with weak local odometric information”. IJCAI, 1997.
[13] Spaan, M.T.J. and Vlassis, N. ”A point-based POMDP algorithm for robot planning”. ICRA, 2004.
[14] Sutton, R. and Barto, A. ”Reinforcement learning; An introduction”. MIT Press, Cambridge, MA. 1998.