APPEARED IN
IEEE International Conference on Robotics & Automation, 2013

Planning How to Learn

Haoyu Bai David Hsu Wee Sun Lee

***Abstract*— When a robot uses an imperfect system model to plan its actions, a key challenge is the *exploration-exploitation* trade-off between two sometimes conflicting objectives: (i) learning and improving the model, and (ii) immediate progress towards the goal, according to the current model. To address model uncertainty systematically, we propose to use Bayesian reinforcement learning and cast it as a partially observable Markov decision process (POMDP). We present a simple algorithm for offline POMDP planning in the continuous state space. Offline planning produces a POMDP policy, which can be executed efficiently online as a finite-state controller. This approach seamlessly integrates planning and learning: it incorporates learning objectives in the computed plan, which then enables the robot to learn nearly optimally online and reach the goal. We evaluated the approach in simulations on two distinct tasks, acrobot swing-up and autonomous vehicle navigation amidst pedestrians, and obtained interesting preliminary results.**

I. INTRODUCTION

Planning under uncertainty is critical for improving the robustness of robotic systems. There has been significant progress recently on robot motion planning algorithms that deal with robot control uncertainty, sensing uncertainty, and environment changes (*e.g.*, [1], [32], [9], [15], [16], [21], [22], [25], [26]). Model uncertainty, however, has received relatively little attention. Consider *acrobot*, a well-studied example in robot control (Fig. 1). If the acrobot's dynamic model is known accurately, we can plan a sequence of actions for the robot to swing up from its natural resting configuration to the stand-up configuration. What shall we do if the dynamic model is imperfect, for example, the link masses or lengths are unknown? Is it worth gathering information to learn and improve the model, at the cost of possibly slowing down immediate progress towards the goal? Or shall we act according to the current model and possibly suffer the consequences of model errors?

This illustrates the classic trade-off between exploration and exploitation, a major challenge in robot planning and learning under uncertainty. It occurs in a wide variety of robotic tasks where we must rely on incomplete or inaccurate models, including, for example, integrated exploration, where a mobile robot navigates through an unknown environment towards a goal, with an imperfect map.

To integrate planning and model learning, we propose to use model-based Bayesian reinforcement learning (RL).

H. Bai, D. Hsu, and W.S. Lee are with the Department of Computer Science, National University of Singapore, Singaore 117417, Singapore.

This work is supported in part by the MoE grant MOE2010-T2-2-071, the SMART IRG grant R-252-000-502-592, and the AOARD grant FA2386-12-1-4031. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

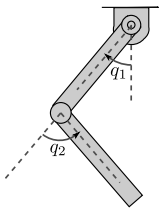

Fig. 1. The acrobot is a two-link articulated robot actuated only at the joint connecting the two links and thus underactuated. It resembles a gymnast swing on a high bar.

Bayesian RL explicitly represents model uncertainty as a probability distribution over the space of all possible models and chooses actions that maximize a robot's expected long-term performance with respect to this distribution.

One approach to Bayesian RL is to cast it as a partially observable Markov decision process (POMDP) [11]. The POMDP state is a pair (s, θ), where s is the robot state and θ represents unknown model parameters. A probability distribution on θ captures the uncertainty in the model parameters. POMDP planning occurs offline. It aims to achieve optimal trade-off between exploration and exploitation by analyzing both aspects of each action: (i) the immediate benefit of moving the robot state towards the goal and (ii) the contribution towards learning the unknown model parameters and reducing model uncertainty. The result is a POMDP policy, which can be represented as a finite-state controller (FSC) and executed very efficiently online. In short, POMDP planning evaluates offline many action plans conditioned on future observations and chooses one that enables the robot to learn as efficiently as possible online and reach the goal.

Example. In the acrobot example, let $s = (q_1, q_2, \dot{q}_1, \dot{q}_2)$ denote the acrobot state, where q_1 and q_2 are joint angles, and $\dot{q}_1$ and $\dot{q}_2$ are angular velocities. The acrobot is controlled by the torque τ at the joint connecting the two links. Applying the standard laws of mechanics, we can derive a dynamic model for the acrobot. Suppose now that the mass m of the second link is unknown in advance. The dynamic model must then depend explicitly on the unknown parameter m:

$$\dot{s} = f(s, \tau; m).$$

The acrobot receives noisy observations of the system state. Our goal is to choose τ that brings up the tip of the acrobot above a specified height.

Standard controller design approaches cannot be easily applied here, as not all model parameters are known. Instead, we treat the entire system as a POMDP and form the augmented state (s, m). We maintain a probability distribution b on (s, m) to capture the uncertainty, in particular, in m

explicitly. We update b by incorporating the observations through Bayesian filtering. A robot action, *i.e.*, the torque applied, may immediately push the acrobot's tip higher towards the goal. At the same time, the action and the ensuing observation lead to an update of b and reduce the uncertainty in m, thus enabling more effective robot actions in the future. Intuitively, when the uncertainty in m is high, the acrobot must act cautiously at the beginning and hedge against different possibilities. As the uncertainty decreases, it then chooses more aggressive actions tailored to specific values of m. POMDP planning analyzes both aspects of each action quantitatively and computes a plan for the acrobot to learn the model parameter and reach the goal simultaneously.

The acrobot must act fast. To meet the demand of efficient online execution, we compute the plan offline and represent the plan as an FSC. The FSC states are labeled with output actions, and the transition edges are labeled with input observations. ◁

This approach provides several key advantages:

- It seamlessly integrates planning and learning in a single unified framework. Offline planning incorporates learning objectives in the computed plan, which then enables nearly optimal online model parameter learning.
- In contrast to standard RL, the model-based Bayesian approach allows us to easily incorporate prior knowledge on the model and reduce the difficulty of online learning. In the acrobot example, although the mass parameter may be unknown, the basic laws of motion and an estimate on the parameter can be used to construct a prior model, leaving online learning the simplified task of reducing uncertainty on the unknown parameter. Such prior knowledge, which is often available for robotic systems, is crucial for efficient online learning.
- This approach is general and flexible. In addition to model uncertainty, it can accommodate control and sensing uncertainty naturally.

One limitation of the approach is the computational complexity of solving POMDPs. In recent years, there have been significant progress in computing approximate solutions to discrete POMDPs [15], [20], [28], [27] as well as more limited progress on continuous POMDPs [3], [24], [22], [31]. These general-purpose algorithms often incur high computational cost for robotic tasks with long planning horizons.

For computational efficiency, we consider a subclass of POMDPs with continuous states, but discrete actions and observations. We also assume deterministic dynamics. For this subclass of models, we provide a simple and effective algorithm that finds approximate solutions. We have evaluated the algorithm in simulation on two distinct tasks, acrobot swing-up and autonomous vehicle navigation amidst pedestrians, and obtained interesting preliminary results.

II. Related Work

Our work addresses the problem of motion planning with model uncertainty. When the model for planning is inaccurate or incomplete, a robot must take action to learn and improve the model. Integrated planning and learning have been studied in various contexts, including model-based RL [10], adaptive dual control [8], integrated exploration [18], and active perception [6]. Our work focuses on model parameter uncertainty and adopts the Bayesian approach to model-based RL. The history of this approach goes back to the work of Bellman [4] and that of Fel'dbaum [7], but efficient algorithms appeared only in recent years.

Several successful algorithms have been proposed for Bayesian RL (*e.g.*, [2], [14]). However, they assume fully observable discrete system states. They also require substantial online computation and are thus not suitable for robotic tasks that require fast response. We argue that the approach of casting a Bayesian RL task as a POMDP [5], [23] and solving for a policy is better suited for a variety of robotic tasks. The offline POMDP policy computation optimally balances model learning and goal achievement. Once a policy is computed, it can be executed efficiently online for robot action selection.

Although solving POMDPs exactly is computationally intractable [19], point-based algorithms have made significant progress in computing approximate solutions. Today the fastest algorithms, such as HSVI [27] and SARSOP [15], can solve discrete POMDPs with hundreds of thousands states in reasonable time. Some recent point-based algorithms deal with continuous state space [3], [22], [33], but their performance on robotic tasks, which often involve long planning horizons, requires further investigation.

The base model that we use is similar to that of earlier work on Bayesian RL [3], [22], [33]. However, none of the earlier algorithms deal with continuous states and long planning horizons together, which are often required for robot motion planning tasks.

III. Problem Formulation

A. Background on POMDPs

Formally a POMDP is a tuple $(S, A, O, T, Z, R, \gamma)$, where S is a set of system states, A is a set of control actions, and O is a set of observations. The transition function $T(s, a, s')$ for $s, s' \in S, a \in A$ specifies the probability of reaching state s' when the system takes action a in state s. The observation function $Z(s', a, o) = p(o|s', a)$ specifies the probability of receiving observation o when the system is in state s'. The reward function $R(s, a)$ specifies the reward received when the system takes action a in state s. Finally, γ is a discount factor that reflects the preference of immediate rewards over future ones.

In a POMDP, the exact state is unknown because of observation noise. The current state is represented as a *belief* $b(s)$, which is a probability distribution over S. We specify an initial belief $b_0(s)$ as an estimate of the initial system state. At each time step, the system executes an action a and receives an observation o. It then updates the belief:

$$b'(s') = \eta Z(s', a, o) \int_{s \in S} b(s) T(s, a, s') \, \mathrm{d}s, \qquad (1)$$

where η is a normalizing factor.

The solution to a POMDP is a *policy* π, which can be represented as a *policy graph* and executed on an FSC. A policy graph has nodes labeled with actions and directed edges labeled with observations. To execute a policy, the FSC starts at an initial node and takes the associated action. It then moves to the next node by following the edge corresponding to the observation received. The process then repeats. The *value* of a policy π is the expected total reward with respect to the initial belief b_0:

$$\mathrm{E}\big(\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\big),$$

where s_t and a_t denote the state and the action at time t, respectively. POMDP planning aims to find an optimal policy π^* with the maximum value.

B. Planning How to Learn

POMDPs can model a variety of robotic systems with control uncertainty, sensing uncertainty, and environment changes. Let $\mathcal{P}$ be the POMDP model of a robotic system. When some system parameters θ are unknown, we denote the model by $\mathcal{P}_\theta$ to indicate the parameter dependence explicitly. $\mathcal{P}_\theta$ is a tuple $(S, A, O, T_\theta, Z_\theta, R_\theta, \gamma)$, where T_θ, Z_θ, and R_θ are parameterized by θ. The set of parametrized POMDP models forms the model space $\mathcal{M} = \{\mathcal{P}_\theta \mid \theta \in \Theta\}$, where Θ is the space of all parameter values. The true model is an element of $\mathcal{M}$, but the controller does not know this true model *a priori* and must learn the parameters while trying to complete the specified task.

We take the approach of Bayesian RL, which captures the model parameter uncertainty explicitly in a probability distribution. Initially, we have a prior distribution over model parameters. It represents our prior knowledge on the model parameters, if there is any. We iteratively update the distribution by incorporating the observations received. Our goal is to compute a policy that enables the robot to complete its task as efficiently as possible, despite the model uncertainty. More precisely, we want to find an optimal policy that maximizes its value with respect to the prior parameter distribution.

Interestingly, our learning problem above can be cast as an augmented POMDP $\mathcal{P}' = (S', A, O, T', Z', R', \gamma)$. The augmented state space $S' = S \times \Theta$ is the cross product of the system state space S and the model parameter space Θ. The action space A and the observation space O of $\mathcal{P}'$ remain unchanged. Since the parameters do not change over time, the transition function T' is defined as

$$\begin{aligned} T'(s,\theta,a,s',\theta') &= p(s',\theta'|s,\theta,a) \\ &= p(s'|s,\theta,a,\theta')p(\theta'|s,\theta,a) \\ &= T_\theta(s,a,s')\delta_{\theta\theta'}, \end{aligned}$$

where the Kronecker delta function $\delta_{\theta\theta'}$ is 1 if $\theta = \theta'$ and 0 otherwise. For the observation function Z' and the reward function R', we have simply $Z'(s',\theta,a,o) = Z_\theta(s',a,o)$ and $R'(s',\theta,a) = R_\theta(s,a)$.

A belief in $\mathcal{P}'$ is a joint probability distribution over $S \times \Theta$. After executing an action a and receiving an observation o, the controller updates the belief:

$$\begin{aligned} b'(s',\theta') &= \tau(b,a,o) \\ &= \eta Z'(s',\theta',a,o)\int_{s\in S,\theta\in\Theta} b(s,\theta)T'(s,\theta,a,s',\theta')\,\mathrm{d}s\mathrm{d}\theta \\ &= \eta Z_{\theta'}(s',a,o)\int_{s\in S} b(s,\theta')T_{\theta'}(s,a,s')\,\mathrm{d}s. \end{aligned} \tag{2}$$

A belief in $\mathcal{P}'$ completely captures the uncertainty in both the system states and the model parameters.

We then perform offline POMDP planning and solve $\mathcal{P}'$ for an optimal or near-optimal policy. Offline planning systematically generates and evaluates many action plans conditioned on future observations. It optimally balances exploration and exploitation by reasoning in the space of beliefs and optimizing the expected total reward. Through this process of offline optimization, we obtain a policy, which can be executed efficiently online on a FSC. The controller learns and adapts to different model parameters and enables the robot to complete the task efficiently.

IV. Algorithm

Here we consider a subclass of POMDPs with continuous states, but discrete actions and observations. We also assume deterministic dynamics. Despite the restrictions, our algorithm is well suited for some quite interesting robotic tasks (Section V), including the acrobot. Simultaneously, we are developing a more general POMDP algorithm that removes many of these restrictions. Results on this new algorithm will be available soon.

To compute a policy for $\mathcal{P}'$ offline, our current algorithm makes use of the QMDP heuristic [17] and leverages the recent advances in motion planning. QMDP is an effective heuristic for solving large discrete POMDPs approximately. It performs a one-step exploration: it reasons about the effect of uncertainty by looking ahead one step and assumes full observability beyond the first step. Specifically, given a belief b, QMDP chooses the action $a^* \in A$ that maximizes the Q-value $Q(b,a)$, which is computed by assuming full observability and solving the Markov decision process (MDP) corresponding to $\mathcal{P}'$. Although the QMDP heuristic may produce suboptimal actions as it does not actively gather information, it often performs well in practice [17].

Solving an MDP with a continuous state space, however, remains difficult. We approximate the Q-value by exploiting our assumption of deterministic system dynamics:

$$Q(b,a) = \int_{s\in S,\theta\in\Theta} b(s,\theta)Q_{\text{MP}}(s,\theta,a)\,\mathrm{d}s\mathrm{d}\theta, \tag{3}$$

where $Q_{\text{MP}}(s,\theta,a)$ is the discounted reward of the best plan for the state s and the parameter value θ starting with the action a. We can compute $Q_{\text{MP}}(s,\theta,a)$ efficiently with any motion planning algorithms, such as PRM [13], RRT* [12], or A* search, as this is a fully deterministic planning problem. In our experiments (Section V), we use a breadth-first search in which the states are clustered into equally spaced bins and each bin is visited only once.

Algorithm 1 Build a policy tree with the QMDP heuristic.

BUILDPOLICYTREE(b)

if all states $s \in S$ with $b(s, \theta) > 0$ for some $\theta \in \Theta$ are terminal **then**
  Create an empty tree T with a single node labeled with NIL action and no children.
else
  $a^* \leftarrow \arg\max_{a \in A} \int_{s \in S, \theta \in \Theta} b(s, \theta) Q_{\text{MP}}(s, \theta, a) \, \mathrm{d}s \, \mathrm{d}\theta$.
  for each $o \in O$ with $p(o|b, a^*) > 0$ **do**
    $b' \leftarrow \tau(b, a^*, o)$.
    $T_o \leftarrow$ BUILDPOLICYTREE(b').
  Create a new tree T whose root node is labeled with action a^*. An outgoing edge of the root is labeled with $o \in O$ and points to the child T_o.
return T.

Algorithm 1 gives an outline of our algorithm. Although the QMDP heuristic appears typically in online planning, our algorithm uses it in the offline setting to construct recursively a policy graph (see Section III), in fact, a policy tree, for a given belief b.

The recursion ends, and an empty tree is returned, when all states $s \in S$ such that $b(s, \theta) > 0$ for some $\theta \in \Theta$ are *terminal*. A terminal state indicates that the robot has either completed the task or failed.

If there is a non-terminal state s and θ with non-zero probability $b(s, \theta)$, we choose the action a^* with the maximum heuristic value $Q(b, a)$, which is computed according to (3) by evaluating each $Q_{\text{MP}}(s, \theta, a)$ on demand with a motion planning algorithm. For each possible observation $o \in O$ with $p(o|b, a^*) > 0$, we then compute a new belief $b' = \tau(b, a^*, o)$ according to (2) and invoke BUILDPOLICYTREE(b') recursively to construct a sub-tree T_o.

We represent a belief as a particle set. The initial belief is obtained by sampling a set of N values from the model parameter space according to the prior $b_0(\theta)$. This effectively creates a sampled model space, and the algorithm operates on this space only. It performs the belief update in (2) through particle filtering and approximates the integral in (3) through Monte-Carlo sampling. The particle belief representation does not require parametric models of $b_0(\theta)$ and the state transition function T, thus simplifying the model construction. The approximation error, as a result of sampling, decreases at the rate $\mathcal{O}(1/\sqrt{N})$ [33].

Each call to BUILDPOLICYTREE(b) constructs one policy tree node by calling the motion planning procedure $\mathcal{O}(N)$ times, as at most N samples represent the belief b. Thus, constructing a policy tree T with $|T|$ nodes requires $\mathcal{O}(N|T|)$ calls to the motion planning procedure. In the worst case, each node in T has $|O|$ children, and $|T|$ grows exponentially with the planning horizon. However, a small policy may exist for many problems. In our setting, system dynamics is deterministic. Once the policy gathers sufficient information, it can act according to the estimated model without further branching on observations. Empirical results in Table I show that the running time and $|T|$ are usually sub-linear to N.

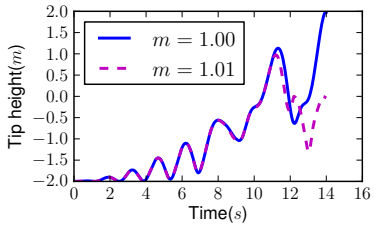

Fig. 2. The acrobot dynamics is sensitive to model parameters. With the same open-loop policy, the acrobot may succeed or fail, as a result of 1% difference in the model parameters.

V. EXPERIMENTS

We experimented with two distinct robotic tasks in simulation: acrobot swing-up and pedestrian avoidance for autonomous vehicles. The results indicate that our approach can potentially be applied to a broad class of robotic tasks.

A. Acrobot

Recall the acrobot example we introduced in Section I. We estimate that the unknown parameter m, the mass of the second link, is in a range $m \in [0.9, 1.1]$. All other parameters are known exactly and the same as those in [30]. We simulate the dynamics with laws of mechanics, and each step of the simulation is $\Delta t = 0.05s$.

Our goal is to swing up the tip to a height above 1.95. The swing-up task is solved as a deterministic motion planning problem if m is known [29]. However, when there is model uncertainty, a conditional control policy is necessary because the system dynamics is very sensitive to the value of m. An open loop policy successfully swinging up for $m = 1.0$ fails for $m = 1.01$ (Fig. 2).

We formulate the acrobot swing-up task with unknown m as Bayesian RL. The prior is an uniform distribution $m \sim U(0.9, 1.1)$. The controller receives a positive reward R when it reaches the goal, and gets a zero reward for the other states and actions. The controller applies torques on the second link of the acrobot, and it can choose a different torque every $10\Delta t = 0.5s$. We restrict the torque to $\tau \in \{+1, 0, -1\}$. The discount factor is $\gamma = 0.95$.

The controller receives discretized values of state $(q_1, q_2, \dot{q}_1, \dot{q}_2)$ as observations. The 4-dimensional continuous states are discretized to equally spaced bins, and each bin is a hyper-cube with the width 0.5. The discretization resembles the limited sensor resolution.

To solve the Bayesian RL POMDP, we sample N parameters from the model prior for the initial belief and computed three POMDP policies with varying N. Each policy is evaluated with $10,000$ simulation episodes with a randomly sampled m for each episode.

Table I shows the performance of POMDP policies with respect to N. We compares POMDP policies with an oracle policy which calculates a control sequence according to the true value of m in each episode. In the table, the second column shows the time spent on offline policy computation; the third column shows the size of the computed policy tree;

TABLE I

COMPARISON OF POLICIES FOR ACROBOT SWING-UP.

Policy	Planning Time (s)	$\|T\|$	Average Height (m)	Average Reward
$N = 1,000$	2,550	4,992	1.799 ± 0.010	0.623 ± 0.006
$N = 10,000$	16,247	33,441	1.830 ± 0.010	0.671 ± 0.005
$N = 50,000$	57,931	109,371	1.877 ± 0.009	0.710 ± 0.004
Oracle	-	-	1.970 ± 0.000	0.756 ± 0.000

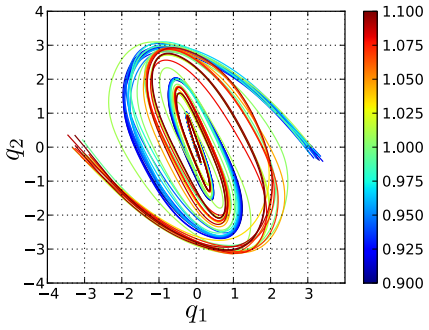

Fig. 3. Acrobot trajectories corresponding to various m values under a same POMDP policy. The colors indicate different m values.

the fourth column shows the final height the tip reached, averaged over all simulations; and the final column shows the average discounted reward. The quality of the POMDP policy improves with an increasing N, as the average reward is getting closer to the oracle policy. This indicates that the POMDP policy can learn m and reach the goal efficiently.

We computed the policies and run simulations on a desktop computer with a 2.83GHz CPU and a 4GB memory. The simulations can be run at 30 KHz for each step, which is sufficient for the requirement of controlling acrobots in real-time.

Fig. 3 shows the behavior of the POMDP policy with $N = 50,000$. It shows 30 trajectories which are generated by using the POMDP policy to control acrobots with m varying from 0.9 to 1.1. For the sake of visualization, we only show (q_1, q_2) changing with time. The trajectories start at $(0, 0)$ and end in regions around either $(\pi, 0)$ or $(-\pi, 0)$, which are the goal regions. These trajectories suggest that the POMDP policy can adapt to different m. When m is low, the acrobot approaches the goal with a counter-clockwise final swing and ends around $(\pi, 0)$. When m is high, it performs one more swing and approaches the goal with a clockwise final swing that ends around $(-\pi, 0)$. The additional swing accounts for the heavier mass of the second link, which needs more energy to reach the same height. We encode the knowledge of system dynamics and model parameter prior into a POMDP model, and solving this POMDP produces a policy that can fully exploit these knowledge. Therefore, when executed online, the policy learns the model quickly, and once its knowledge of model parameters become sufficient, it can complete the task efficiently.

Fig. 4 shows the behavior of a POMDP policy at different uncertainty levels. It shows the average belief entropy and the variance of the torque τ at time t. The results are collected from 100 simulations. The entropy reduces over time as the POMDP policy receives more and more observations. The

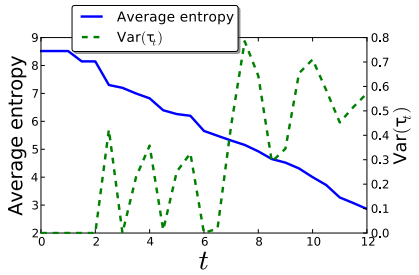

Fig. 4. The average belief entropy and the torque variance over time for a POMDP policy in simulation.

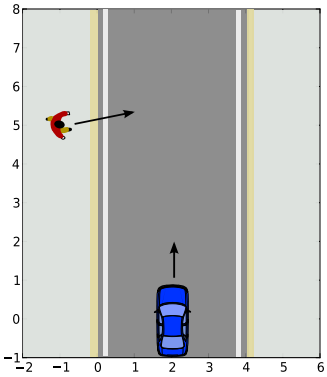

Fig. 5. Pedestrian avoidance.

variance of τ is zero at the beginning, reflecting that the policy has no information about m and always applies the same torques. The variance of τ increases as the entropy decreases, because the policy tends to apply torques more specific to the current parameter estimation whenever it is necessary. Therefore, after acquiring new knowledge with online learning, the policy can exploit them to improve its performance.

For comparison, we tried earlier Bayesian RL algorithms on this task. None of the online algorithms (*e.g.*, [2], [14]) can be applied here, because they cannot meet the real-time requirement of acrobot control. The offline algorithms [23], [33] require discrete POMDP models. The acrobot dynamics is, however, very sensitive (see Fig. 2). A coarse discretization does not capture the main dynamic features, and a fine discretization generates so many states, which are beyond even the fastest discrete POMDP algorithms.

B. Pedestrian Avoidance

In the pedestrian avoidance task, an autonomous vehicle driving on the road should avoid a pedestrian who is also crossing the road (Fig. 5). The vehicle has relatively precise control for itself, but does not know the pedestrian's intention, which we treat as an unknown model parameter. To ensure driving safety while avoiding unnecessary stops, the vehicle has to learn the pedestrian's intention from noisy observations of her movements.

We model the vehicle's state by its position and speed. Position y_0 is the distance traveled by the vehicle from time $t = 0$. The vehicle's speed v_0 is ranging from 0 m/s (full stop) to 2 m/s. At every time step, the vehicle can take

TABLE II

COMPARISON OF POLICIES FOR PEDESTRIAN AVOIDANCE.

Policy	Planning Time (s)	$\|T\|$	Time	Accident
$C = 1$	163	1401	6.74	0.000957
$C = 10$	165	1416	6.74	0.000054
$C = 100$	162	1441	6.77	0.0
Oracle	-	-	6.42	0.0
Bayes averaging	-	-	6.55	0.442

one of the three actions to control its speed: ACCELERATE ($1\ m/s^2$), MAINTAIN ($0\ m/s^2$) and DECELERATE ($-1\ m/s^2$). The pedestrian's position is (x_1, y_1) and her intention is modeled by the walking direction $\phi \in [-\pi/12, \pi/12]$ and the speed $v_1 \in [0.8, 1.2]$. The vehicle cannot directly observe ϕ and v_1. The only observation is the pedestrian's position, and is discretized to $0.5m \times 0.5m$ grids, which resembles a limited sensor resolution.

At each episode, the vehicle's initial state is $y_0 = 0$ and $v_0 = 2.0$; the pedestrian's initial position is $(x_1, y_1) = (-1.0, 5.0)$ but her intention is randomly drawn from an uniform prior: $\phi \sim U(-\pi/12, \pi/12)$ and $v_1 \sim U(0.8, 1.2)$. An episode will finish when the vehicle successfully drives to $y_0 = 7.0$ or a collision happens. The vehicle will get a positive reward $+1$ when it succeeds, and a negative reward $-C$ for collisions. The rewards are discounted with $\gamma = 0.95$. Each time step has a duration of 0.5 seconds.

We model the task as a continuous-state POMDP and use our algorithm to solve it. We choose the number of sampled model parameters $N = 1,000,000$ and compute policies for three POMDP models with different collision costs: $C = 1$, $C = 10$ and $C = 100$.

We compare the computed POMDP policies with the oracle policy and the Bayes model averaging policy. The oracle policy always chooses the best action with respect to the pedestrian's true intention. The Bayes averaging policy monitors beliefs online and calculates the expected state $\bar{s} = \mathrm{E}(s)$ with respect to the current belief b. It then chooses the best action for $\bar{s}$ by querying a motion planner. POMDP policies are evaluated with $1,000,000$ simulations. The oracle and Bayes averaging policies are evaluated with $1,000$ simulations due to their higher computational cost as online algorithms.

In Table II, the second column shows the time spent on offline planning; the third column shows the size of the policy tree; the fourth column shows the averaged time for the vehicle to pass the road; and the last column shows the accident rate in simulations. With a larger collision cost, the POMDP policy becomes more conservative: it passes the road slower and gets a lower accident rate. With $C = 100$, it achieves a zero accident rate, which is comparable to the oracle policy in terms of safety. POMDP policies drive a little slower than the oracle policy, because to ensure safety, it takes more time to estimate the pedestrian's intention. The Bayes averaging policy results a much higher accident rate.

Why does the POMDP policy perform as well as the oracle policy and much better than the Bayes averaging policy? Fig. 6 shows the average belief entropy profile of POMDP policy, and the speed profiles of the POMDP, oracle, and Bayes averaging policy. The results are obtained from 200 simulations. These profiles show the change of uncertainty and speed over time for each policy. All policies decelerate to nearly a full stop to wait the pedestrian to pass. But the POMDP policy slows down earlier than the oracle policy to hedge against uncertainty. Before time $t = 2$, the entropy profile indicates a high level of uncertainty, so the POMDP policy decelerates faster than the oracle policy. After $t = 2$, the entropy decreases quickly and the POMDP policy begins to accelerate. As the entropy decreases, the POMDP policy further accelerates and eventually reaches a speed similar to the oracle policy. The Bayes averaging policy and oracle policy have similar speed profiles, because both policies are planning against a single state at each time step: the ground truth state for the oracle policy and the averaged state $\bar{s}$ for the Bayes averaging policy. However, $\bar{s}$ does not represent full information as in the belief. By planning against $\bar{s}$, the Bayes averaging policy neglects some states that possibly lead to collisions, thus resulting in a high collision rate. Therefore, planning in the belief space is necessary for learning the pedestrian's intention and avoiding collisions.

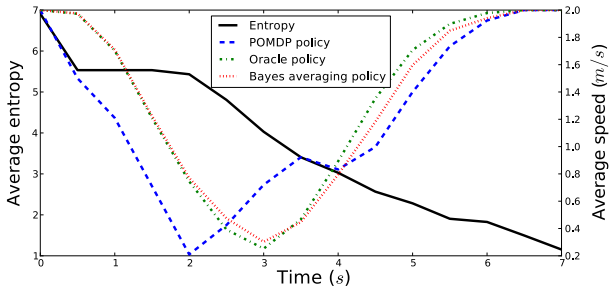

Fig. 6. The average belief entropy and the average vehicle speed for pedestrian avoidance in simulations.

VI. CONCLUSION

We treat robot motion planning under model uncertainty as Bayesian reinforcement learning and solve it by constructing an augmented POMDP over the joint space of robot states and model parameter values. This approach integrates planning and learning seamlessly: offline POMDP planning incorporates learning objectives in the computed plan, which then enables a robot to learn the true model online and complete the task efficiently.

A large number of unknown parameters may pose difficulty for offline POMDP planning, as both the planning time and the size of resulting policies grow. In this case, online planning can help reduce computational complexity. In the future, we will investigate ways to bring together the benefits of offline and online planning.

Acknowledgments. This work is supported in part by MoE AcRF grant 2010-T2-2-071 and National Research Foundation Singapore through the Singapore MIT Alliance for Research and Technology's Future Urban Mobility IRG research program.

REFERENCES

[1] R. Alterovitz, T. T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Proc. Robotics: Science and Systems*, 2007.

[2] J. Asmuth, L. Li, M. Littman, A. Nouri, and D. Wingate, "A bayesian sampling approach to exploration in reinforcement learning," in *Proc. Uncertainty in Artificial Intelligence*, 2009, pp. 19–26.

[3] H. Bai, D. Hsu, W. Lee, and V. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," in *Algorithmic Foundations of Robotics IX—Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, D. Hsu *et al.*, Eds. Springer, 2010.

[4] R. Bellman and R. Kalaba, "On adaptive control processes," *IRE Trans. on Automatic Control*, vol. 4, no. 2, pp. 1–9, 1959.

[5] M. Duff, "Optimal learning: Computational procedures for bayes-adaptive Markov decision processes," Ph.D. dissertation, University of Massachusetts Amherst, 2002.

[6] S. Dutta Roy, S. Chaudhury, and S. Banerjee, "Active recognition through next view planning: a survey," *Pattern Recognition*, vol. 37, no. 3, pp. 429–446, 2004.

[7] A. Fel'dbaum, *Optimal Control Systems*. Academic Press, 1965.

[8] N. Filatov and H. Unbehauen, *Adaptive Dual Control: Theory and Applications*. Springer, 2004.

[9] T. Fraichard and R. Mermond, "Path planning with uncertainty for car-like robots," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 1998.

[10] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *J. Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[11] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] L. Kavraki, P. Švestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] J. Kolter and A. Ng, "Near-bayesian exploration in polynomial time," in *Proc. Int. Conf. on Machine Learning*, 2009.

[15] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proc. Robotics: Science and Systems*, 2008.

[16] S. LaValle and S. Hutchinson, "An objective-based framework for motion planning under sensing and control uncertainties," *Int. J. Robotics Research*, vol. 17, no. 1, pp. 19–42, 1998.

[17] M. Littman, A. Cassandra, and L. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Proc. Int. Conf. on Machine Learning*, 1995.

[18] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2002.

[19] C. Papadimitriou and J. Tsisiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[20] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. Int. Jnt. Conf. on Artificial Intelligence*, 2003, pp. 477–484.

[21] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Pérez, "Belief space planning assuming maximum likelihood observations," in *Proc. Robotics: Science and Systems*, 2010.

[22] J. Porta, N. Vlassis, M. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *J. Machine Learning Research*, vol. 7, pp. 2329–2367, 2006.

[23] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proc. Int. Conf. on Machine Learning*, 2006.

[24] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance," in *Proc. Int. Symp. on Robotics Research*, 2007.

[25] N. Roy and S. Prentice, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Robotics Research*, vol. 28, no. 11–12, pp. 1448–1465, 2009.

[26] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Information Processing Systems (NIPS)*. The MIT Press, 1999, vol. 12, pp. 1043–1049.

[27] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. Uncertainty in Artificial Intelligence*, 2005.

[28] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[29] M. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 49–55, 1995.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning*. MIT Press, 1998.

[31] S. Thrun, "Monte Carlo POMDPs," in *Advances in Neural Information Processing Systems (NIPS)*. The MIT Press, 2000.

[32] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," in *Proc. Robotics: Science and Systems*, 2010.

[33] Y. Wang, K. Won, D. Hsu, and W. Lee, "Monte Carlo Bayesian reinforcement learning," in *Proc. Int. Conf. on Machine Learning*, 2012.