CS674 - Spring 2016 - Paper Presentation

Chaitanya Mitash, Zacharias Psarakis

May 11, 2016

1 POMDP Planning for Robust Robot Control

1.1 Introduction

The Partially Observable Markov Decision Process offers a framework for real-world planning and control problems. The main challenge when using a POMDP is that the world states are not directly observable, which makes the process of coming up with an exact solution intractable because of the following reason. POMDPs assume that the agent cannot observe directly the world states. Instead, a partial knowledge of the world state is available - the *belief state*. The space of *information states* is defined as the space of all belief states of the world. The problem grows exponentially depending on the available *planning horizon*.

In order to overcome this issue *point-based techniques* have been developed. A point-based technique, instead of arguing over the whole set of information states the sample in certain ways the "best" beliefs and compute an approximate solution using these samples. The solution provided from these samples is not guaranteed to be the optimal solution and the quality depends on which belief states were selected.

The contribution of this paper is an introduction of a new point-based value approximation over a set of beliefs that minimize a bound on the error of the value approximation.

1.2 Background and Terminology

The main aim of Partially Observable Markov Decision Process is to provide a *policy* that will guide the agent to select the optimal *action* depending on the current partial observation of the state of the world.

A POMDP is defined be the n-tuple S, A, Z, b_0, T, O, R where:

- S defines the set of States
- A defines the set of available to the agent actions
- Z defines the set of observations
- b_0 describes the probability that the world is at each of the states in S at time $t = 0$
- $T(s, a, s')$ is the transition function. The transition function defines the probability that the world will transit from state s to state s' when action a is performed by the agent
- $O(s, a, z)$ describes the probability that the agent will observe z when from state s performs the action a
- $R(s, a)$ is the reward function that describes the reward the agent will get if action a is performed while in state s

As stated previously the agent has partial information about the state of the world. This means that the state s_t is not directly observable. Instead a set of partial observations are available to the agent $z_1, z_2, ..., z_t$. From these partial observations the agent can approximate a *belief*. A belief is a probability that the agent estimates the state of the world is equal to s given all the observations and actions until time t: $b_t(s) = Pr(s_t = s | z_t, a_{t-1}, ..., a_0)$.

1.3 Belief Computation

In many robotics applications the belief estimation can be computationally challenging as the current belief depends on all previous observations and beliefs. However, because POMDPs are instance of a Markov processes the belief b_t at time t can be computed recursively using only the belief of the previous time step b_{t-1}, the last action the agent performed a_{t-1} and the observation z_t:

$$b_t(s) = \tau(b_{t-1}, a_{t-1}, z_t) := \frac{\sum_{s'} O(s', a_{t-1}, z_t) T(s, a_{t-1}, s') b_{t-1}(s')}{Pr(z_t | b_{t-1}, a_{t-1})}. \tag{1}$$

1.4 Policy Computation

The estimation of the belief state can be computationally challenging, however in this paper the research is mostly focused on how to identify the best policy that an agent can use. This means that the belief state estimation is considered to be computed accurately and the reasoning will focus on how to compute good policies.

A policy maps a specific belief state to an optimal action the agent will choose to take: $\pi(b) \rightarrow a$, where b is a belief distribution and a is the action chosen by the policy π. A policy $\pi^*(b_t)$ is considered to be optimal when the discounted future reward is maximized:

$$\pi^*(b_t) = argmax_\pi E_\pi [\sum_{t=t_0}^{T} \gamma^{t-t_0} r_t | b_t]. \tag{2}$$

This policy has to be computed over all the possible beliefs which is a challenging procedure [5] and many recent research publications propose applying value updates in a few specific belief states [6] rather than the whole set of belief states. All the proposed techniques differ in how they choose the belief point to update out of the set of belief points, but the procedure they follow to update the values is the same.

1.4.1 Point-Based Value Update

Consider a set of belief points $B = \{b_0, b_1, ..., b_q\}$. For each of these belief points we define an α-vector which contains an estimate of the value function. From all these vector we generate a new set $\Gamma_t = \{\alpha_0, \alpha_1, ..., \alpha_q\}$. In order to perform a point-based value update we define two intermediate sets [6] $\Gamma_t^{a,*}$ and $\Gamma_t^{a,z}$, $\forall a \in A, \forall z \in Z$:

$$\Gamma_t^{a,*} \leftarrow \{\alpha^{a,*}\}, \text{ where } \alpha^{a,*} = R(s, a) \tag{3}$$

$$\Gamma_t^{a,z} \leftarrow \{\alpha^{a,z} | \alpha_i \exists \Gamma_{t-1}\}, \text{ where } \alpha_i^{a,z} = \gamma \sum_{s' \in S} T(s, a, s') O(s', a, z) \alpha_i(s'). \tag{4}$$

The first intermediate set contains the immediate rewards of performing an action a while in state s, while the second set contains the expected future discounted rewards as defined in Bellman's equation (Bellman, R. E. 1957). Now we take the expectations over the observations and construct a set $\Gamma_t^b, \forall b \in B$:

$$\Gamma_t^b \leftarrow \{\alpha^{a,b} | a \in A\} \text{ where } \alpha^{a,b} = \Gamma_t^{a,*} + \sum_{z \in Z} argmax_{\alpha \in \Gamma_t^{a,z}} \sum_{s \in S} \alpha(s) b(s). \tag{5}$$

Set Γ_t^b contains the α-vectors that maximize the immediate plus the expected reward over all the possible expected future discounted rewards for each belief state. Now we can compute the best possible action for each belief point:

$$\Gamma_t \leftarrow \{\alpha^b | b \in B\}, \text{ where } \alpha^b = argmax_{\alpha \in Gamma_t^b} \sum_s \alpha(s) b(s) \tag{6}$$

The computational time of the above set is polynomial since the size of the set is constant. Notice here that while the set preserves only the best α-vector at each of the belief points, we can easily get an estimate of the value function in every point of the belief state:

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s) b(s). \tag{7}$$

1.5 Error Bound on Point-Based Value Updates

A bound for multiple value updates is defined in [6] as:

$$\|V_T^B - V_T^*\|_\infty \leq \frac{(R_{max} - R_{min}) \max_{b' \in \Delta} \min_{b \in B} \|b - b'\|_1}{(1-\gamma)^2}, \tag{8}$$

where $b' \in \Delta$ is the belief point that is contained in set B that the point-based update makes its worse error and $b \in B$ is the closest sampled belief point to b'. Intuitively, the above inequality demonstrates that the error is bounded by the worst update that is done when using the closest to the current belief state, sampled belief point.

1.6 Error-Minimization Point Selection

In this section we will describe the contribution of this paper. The paper proposes an algorithm for selecting good belief points out of the set of belief states S. The algorithm uses equation 8 in order to select those points that mostly reduce the error bound. Starting from b_0 a tree is built combining all possible actions a that will lead to a new belief b_k and possible new observations z that the agent will sense at the new belief state b_k.

If point-based value updates are performed to *all* of the nodes of the tree, then optimal performance is guaranteed. However the computational cost explodes. Instead of updating all possible beliefs, only the beliefs that most likely will minimize the error will be updated.

Let us consider 3 sets of nodes:

- set B contains all the already selected belief points
- set $\hat{B}$ contains all the candidate belief points. The algorithm removes belief points from that set and adds them to set B
- the last set contains all other reachable belief points

Every point added in set B should improve the estimate of the value function as much as possible. To find these points the analysis of the previous section is used. The innovation of the algorithm proposed is that instead of always choosing the point with the higher error bound, the belief point that is most likely to occur *while* maximizing the error bound is selected. This means that the knowledge of the observation probability distribution is used when selecting a new belief point. The belief point that maximizes the following equation is chosen.

$$\hat{\epsilon}(b) = \max_{a \in A} \sum_{z \in Z} O(b, a, z) \epsilon(\tau(b, a, z)), \tag{9}$$

where $\epsilon(\tau(b, a, z)) \leq \alpha' b' - \alpha b'$, α is the vector that is maximal at b and α' is the vector that would be maximal at b'. In order for a belief point to be chosen and added to set B, equation 9 is used to find an existing point in B with the largest error bound, and then its descendant that has the greater impact on $\hat{e}(b)$ is chosen and added to set B.

The results of the application of the above algorithms are presented in the rest of the paper.

2 Simultaneous Localization and Grasping as a Belief Space Control Problem

2.1 Introduction

The problem of generating feasible plans for grasping objects at a defined pose is a well-studied problem [1, 2], however this is rarely the case in real world scenarios as sensing is expected to have some noise. Thus the more challenging task is to simultaneously localize and grasp objects in a difficult to perceive configuration. This implies that we start with an uncertainity or belief of the state of the object with the objective to compute a control to reach a belief state where the probability that the object has been successfully grasped is high.

One way to approach the problem of grasping under uncertainity is to frame it as a partially observable Markov decision process(POMDP) and use point-based POMDP solvers [3] to get polynomial time approximations to the optimal policy. This paper however, discusses a new hypothesis based belief space control algorithm [4] to solve the problem which is effective also in non-Gaussian belief spaces.

2.2 Problem Statement

Following are the set of definitions required for understanding the method :
Grasp State : The grasp state may include the pose and shape of the object to be grasped, obstacles, potential occlusions. Let it be denoted by
$x \in R^n$
Process Dynamics : Let $u \in R^l$ be an action vector, the process dynamics can be defined as as deterministic and non-linear function of action and current state
$x_{t+1} = f(x_t, u_t)$
Observation model : the system must infer grasp states based on observations, $z \in R^m$, made at each time step according to a non-linear and stochastic observation function,
$z_t = h(x_t) + v_t$, where v_t is gaussian noise and h describes expected observation as a function of grasp state.
Target grasp configurations : $g \subset R^n$ identified by the system designer

We track the probability density function $\pi(x, b)$ over grasp state, where $b \in B$ is the belief state. This is updated over time as a function of action and observation using bayes filtering as,
$\pi(f(x, u_t), b_{t+1}) = \frac{\pi(x,b_t)P(z_{t+1}|x,u_t)}{P(z_{t+1})}$

The problem reduces to : starting from $\pi(x, b_1)$ the problem is to reach a configuration where posterior distribution $\pi(x, b_T)$ is very peaked about grasp configuration G.

2.3 Objectives

- Given a prior belief state, we define a "hypothesis" state at the maximum of the distribution as $x^1 = argmax_{x \in R^n} \pi(x; b_1)$ and also sample (k-1) other states x^i such that $\pi(x^i, b_1) \geq \phi$
- We create a horizon T belief space plan that confirms or disproves the hypothesis state. A re-planning step is triggered when during plan execution, the true belief state diverges too far from the nominal trajectory.
- To compute a sequence of control $u_1, u_2, ...u_{T-1}$, such that the following condition is true : $\theta(b_T, G) = \int_{x \in G} \pi(x; b_T) \geq \omega$, where $\pi(x; b_T)$ denotes the posterior probability distribution over grasp state.
- the control sequence should minimize the probability of seeing the observation sequence expected in the sample states when the system is in hypothesis state.

We use trajectory optimization to solve the planning problem to find a path through the parameter space. Sequential quadratic programming is used to solve the problem for samples x^1, x^k, goal constraint x_g and time horizon T.

Sometimes, the problem is solved without the goal constraint. The planner is represented as :
$u_{1:T-1} = DIRTRAIN(x^1, ...x^k, G, T)$ or $u_{1:T-1} = DIRTRAIN(x^1, ...x^k, T)$

2.4 PseudoCode

Algorithm 1: Belief-space re-planning algorithm

Data: Initial belief b, goal region G, planning horizon T
1 **while** $\theta(b, G) < \omega$ **do**
2 $x^1 \leftarrow$ new hypothesis state;
3 sample (k-1) states,;
4 $b'_{1:T}, u_{1:T-1} = CreatePlan(b, x^1, ..x^k, x_g, T)$;
5 **for** *i ∈ 1 to T-1* **do**
6 Update belief b_{t+1}, executing u_t and perceive observation z_{t+1};
7 **if** *dist*$(\pi(x, b_{t+1}), \pi(x, b'_{t+1}) > \theta$ **then**
8 break;

Algorithm 2: CreatePlan

Data: Initial belief b, goal region G, planning horizon T
Result: nominal trajectory $b_{1:T-1}$ and $u_{1:T-1}$
1 $u_{1:T-1} = DIRTRAIN(x^1, ...x^k, G, T)$;
2 Update belief b'_{t+1} based on the expected observation h;
3 **if** $\theta(b, G) < \omega$ **then**
4 $u_{1:T-1} = DIRTRAIN(x^1, ...x^k, T)$;
5 Update belief b'_{t+1} based on the expected observation h;

- The re-planning algorithm runs until the belief of being in goal grasping configuration exceeds a threshold ω
- Hypothesis and the sample states are re-generated and CreatePlan method is invoked to compute a path to reach goal configuration while minimizing the probability around sampled states.
- In the function CreatePlan, in case a plan does not exist to reach the goal state with high probability in step 1, we try to compute a plan without the goal configuration but with the objective to localize.
- The belief state returned by create plan is generated by the computing the expected observation following the trajectory given by the plan, starting from the initial belief.
- In line 7 of the re-planning algorithm we compare the above belief state with the actual belief state generated by using the measurement updates in each time step.

2.5 Experimental Settings

There were two experimental settings discussed in this paper. In the first setting we use a horizontal-pointing laser mounted to the end-effector of a two link robot arm. It moves along the vertical axis and measures range from the end effector to what it preceives. The objective was to localize the position of the end effector and to reach a goal state corresponding to a gap between two vertically mounted boxes. The paper describes an iteration of the method where the hypothesis is disproved due to the distance between the actual belief state and the nominal trajectory obtained from hypothesis.

In the second setting, a multi arm robot is used to localize the pose and dimension of boxes using laser scanner and also be able to grasp the boxes. The method was able to converge to solutions within minimal error distances under difficult settings where boxes were pressed against each other.

References

[1] D. Berenson, S. Srinivasa, and J. Kuffner. Task space regions: A framework for poseconstrained manipulation planning. Intl Journal of Robotics Research, March 2011.

[2] V. Nguyen. Constructing stable grasps in 3d. In IEEE Intl Conf. Robotics Automation, volume 4, pages 234239, March 1987.

[3] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In Proceedings of Robotics: Science and Systems (RSS), 2008.

[4] R. Platt, L. Kaelbling, T. Lozano-Perez, and R. Tedrake. A hypothesis-based algorithm for planning and control in non-gaussian belief spaces. Technical report, Massachusetts Institute of Technology, 2011.

[5] Anthony R. Cassandra, Michael L. Littman, Nevin Lianwen Zhang. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI), pages 54-61, 1997

[6] J.Pineau, G. Gordon ans S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In Proceedings of IJCAI 2003