Implementation of Partially Observable Markov Decision Process in Robotics

Gyanig Kumar
Computer Science
University of Colorado
gyanig.kumar@colorado.edu

Saksham Khatwani
Computer Science
University of Colorado
saksham.khatwani@colorado.edu

Uditanshi Tomar
Computer Science
University of Colorado
uditanshu.tomar@colorado.edu

***Abstract*—The area of robotics deals with highly uncertain probability in terms of creating a real life operation based on a simulation. The probability of determining a goal in a dynamic environment is one of the most interesting problems in robotics. Various solver algorithms are generally highly computation intensive which can be for path planning or robot joint state estimation. In regards to learning from the probabilistic learning, we explore the domain of Partially Observable Markov Decision Processes(POMDP) and created a simulation to work on the integration of POMDPs. The task that we found interesting is learning from human intent i.e keyboard inputs to control a robotic manipulator for picking objects. We observed that the human intent is a reliable source of guiding for POMDP solvers. The simulation software used is a MuJuCo environment and we used Julia POMDP packages and tools to integrate them together with Robot Operating System(ROS). Our approach utilizes the human input to improve the robot arm movement both in 2D space and 3D space initialization. The area of POMDP is highly interesting as it deals with dynamically environment very efficiently.**

I. Introduction

A. Partially Observable Markov Decision Process

Stochastic processes are all around us. They are useful in modeling any time series process. Markov decision process deals with moving from one state to another in stochastic environments [Wiering and M. van Otterlo(2012)]. While in normal Markov decision processes, we are aware of the current state that we are in and we have the transition matrix to tell us how to move to the next state.

Partially Observable Markov Decision Processes **(POMDPs)** are Markov Decision Process with an added complexity of not knowing the current state that we are in. In POMDPs, we have an idea or approximation of the state space, i.e. we have a probability distribution that represents the probability of an agent being in a particular state. This is known as **Belief**.

For a POMDP system, we have a set of **Actions** that we can take in order to move towards a given goal state. Each action can result in some **Observation** and can also yield some **Reward**. Any agent in a POMDP system would like to take actions such that it yields the highest rewards. The observations can be used by the agent to have an informed opinion about what action to take next. Meaning, the observations help the agent to become more certain about their belief, enabling the agent to take actions more confidently.

So more formally, we can say that a partially observable Markov Decision Process can be represented with a tuple S, A, Ω, T, O, R where **S** is a finite set of states, **A** is a finite set of actions, Ω is a finite set of observations, **T** is a transition function, **O** is an observation function and **R** is a reward function.

B. POMDP Solvers

Given the POMDP problems require agents to account for a lot of Action-Observation pairs, it becomes quite computationally expensive in real world scenarios such as robotics. To mitigate this, there are many solvers that we can use to optimally solve the POMDP problem. [Lauri et al.(2023)Lauri, Hsu, and Pajarinen] talks about the various kinds of solvers that are popular in robotics. Here are a few which can be used for the object grasping task:

1) Point Based Value Iteration: Continuous high-dimensional markov state are quite complicated to solve. The Point Based Value Iteration is an approximate algorithm, which which tries to reduce the dimensionality of the belief state by having a small set of belief points. These points are bounded by certain heuristics based on the use case. The boundation of the belief points leads to computational savings. The authors of [Hsiao et al.(2007)Hsiao, Kaelbling, and Lozano-Perez] utilize this method to achieve the robotic arm grasping task.

2) Monte Carlo Tree Search: Given the POMDP problems are guided by a set of possible Action-Observation states, they can be thought of as a tree where the Action and Observation states are the nodes. Monte-Carlo Tree search is essentially, is a black-box generator model where the solver starts from a particular belief state and generates the possible tree structure. It selects a particular node based on the heuristics such as rewards, and it tries to go down the path which could yield the highest reward. We can choose to explore the tree until a certain depth based on our computation.

[Katt et al.(2018)Katt, Oliehoek, and Amato] and [Xiao et al.(2019)Xiao, Katt, Pas, Chen, and Amato] are a few ap-

proaches that use MCTS planning. In [Xiao et al.(2019)Xiao, Katt, Pas, Chen, and Amato], MCTS is used for both target searching and grasping.

II. Research Questions

Markov decision processes are highly computation intensive as they observe and control the entire state space along with the transition probability. And, it becomes computationally expensive if we consider 3D space for our state-transition-observation matrix. We wanted to explore the accuracy of POMDPs in the 3D space and test the computation load. POMDPs are independent of training but require greedy solvers' methods like Monte Carlo Tree Search or point-based value iterations. On the basis of POMDPs in 3D space, we explored the use of human intent i.e moving the robot to predict the intent of the human in our experiment. In the following sections, the paper explains the environment design and requirements for the experimentation.

III. Methodologies

In order to evaluate the performance of our POMDP based system, we used the MuJoCo (Multi joint dynamics with contact) physics engine, a high fidelity simulation environment for continuous control tasks. It was selected for its seamless simulation of rigid body dynamics. The robotic platform that we used in the simulator is the Sawyer robotic arm, a 7 degree of freedom robot with torque control manipulator and a gripper arm end. It was selected for its precision, flexibility and support for Cartesian-space control. Our setup initialized the robot to move the gripper in x,y and z axis and interact with different objects on the table. The assigned task is to reach on the top of the object, which is not directly disclosed to the robot. This object can also be changed by human inputs by the user. The core idea is to simulate human-robot interaction under uncertainty and how it impacts the end behavior of the robot. In our setup, three objects are placed on the table. A bowl, spoon and a lemon. The initial belief is set to bowl (initial goal position). If the robot's operations go unhindered the robot gripper is supposed to navigate to the bowl unassisted with no prior knowledge of the arm trajectory and the local of the bowl. If user input is given which makes the arm move to a new object (in our case lemon or spoon) the belief distribution gradually changes and a new target object is selected by the robot to navigate to.

A. POMDP Intialization

The `SAPOMDP` constructor function maps an input collection of objects, denoted $\mathcal{O}$, to a Partially Observable Markov Decision Process (POMDP) model. This mapping is mathematically expressed as:

$$\text{SAPOMDP} : \mathcal{O} \mapsto (\mathcal{E}, N, \gamma, \Delta t)$$

where:

- $\mathcal{O} = \{o_1, o_2, \ldots, o_N\}$ is the set of objects, with each $o_i = [x_i, y_i, z_i] \in \mathbb{R}^3$ representing a 3D position.
- $\mathcal{E}$ is the environment, defined as a tuple:

  $$\mathcal{E} = \big([0.0, 2.0]\times[-1.0, 1.0]\times[0.0, 2.0], \emptyset, \{[x_i, y_i, z_i]\}_{i=1}^{N}\big)$$

  where:
  - $[0.0, 2.0] \times [-1.0, 1.0] \times [0.0, 2.0]$ represents the workspace boundaries for the x, y, and z coordinates, respectively.
  - $\emptyset$ denotes an empty set of static obstacles.
  - $\{[x_i, y_i, z_i]\}_{i=1}^{N}$ is the set of goal locations derived from the input objects.
- $N = |\mathcal{O}|$ is the number of objects in the input collection.
- $\gamma = 0.7$ is the discount factor for future rewards.
- $\Delta t = 0.1$ is the time step for the POMDP dynamics.

The belief state is represented as a probability distribution over a set of possible goals. Formally, the belief is defined as:

$$b = \{(id_i, name_i, \mathbf{p}_i, \pi_i)\}_{i=1}^{N},$$

where:

- $id_i \in \mathbb{Z}$ is the unique identifier for goal i,
- $name_i \in \mathcal{S}$ is the name of goal i,
- $\mathbf{p}_i = [x_i, y_i, z_i]^\top \in \mathbb{R}^3$ is the 3D position of goal i,
- $\pi_i \in [0, 1]$ is the probability that goal i is the intended target, with $\sum_{i=1}^{N} \pi_i = 1$.

The cost of a trajectory to goal i is defined as the sum of the Euclidean distance from the current position to the next position and from the next position to the goal:

$$C_i = \|\mathbf{r}_{t+1} - \mathbf{r}_t\|_2 + \|\mathbf{p}_i - \mathbf{r}_{t+1}\|_2.$$

The belief probability for goal i is updated using a Boltzmann distribution:

$$\pi_i' = \pi_i \cdot \exp\left(-\frac{C_i}{\beta}\right),$$

where π_i is the prior probability. To ensure the probabilities sum to one, normalization is applied:

$$\pi_i = \frac{\pi_i'}{\sum_{j=1}^{N} \pi_j'}.$$

This update increases the probability of goals whose trajectories are made more efficient by the robot's movement.

B. Use of Intent Learning

Understanding human intent is really crucial in the future of collaborative robotics. We are interested in exploring if the robotic agent is able to account for human intent. In our experiment, we have a robotic arm with a default target object to grasp. Which means that by default, it will plan its action in order for the default object $bowlinourcase$ to reach the highest belief value.

We want to test if human input can act as meaningful "observations" to guide the robotic arm to move towards the object that the human intends it to. We model intent as a variable that is hidden in POMDP state spaces. The robot does not observe the intent of the user directly but receives the observations that are correlated with it. Key aspects of human intent learning are:

1) Sequential signal interpretation - Observations are evaluated as history, which enables context aware interpretation.
2) Belief sharpening - The belief concentrates to the most likely goal when more consistent input is observed, reducing entropy.
3) Handling ambiguity - If the input are inconsistent, the belief distribution remains broad and action commitment is delayed.
4) Feedback loop - The robot's movement provokes more human input, hence creating a closed loop interaction.

C. Metrics

1) Belief State Distribution: In POMDP, the agent cannot observe the true state. So instead, it maintains a categorical probability vector b, where each component $b_k = P(s_k \mid \text{history})$ represents the probability of a system being in state s_k given current and past observations. It can be denoted as:

$$b_{t+1}(s') = \eta\, \mathcal{O}(o_{t+1} \mid s', a_t) \sum_{s \in \mathcal{S}} \mathcal{T}(s' \mid s, o_{t+1}) b_t(s)$$

Every single decision that the agent makes is based on this evolving vector b. If the belief is sharp, then the robot is confident. And if it is flat, the robot has no idea where it is. The belief is updated at every step using Bayesian belief updating .

In the simulation, we initialized the belief as a uniform distribution over the relevant hidden state components and then repeatedly applied the POMDP update equation to propagate it. This belief state is the core internal state for the operation of the POMDP solver. All the decision making, ultimately, depends on it rather than the true state of robot. In our robotic grasp task, if the hidden state is the target that the human decides to choose and the belief bt will assign probabilities to bowl, spoon and lemon; referring to the likelihood that each object is the intended target at time t. As more human inputs (observations) are provided, the probability mass in b gets concentrated to the correct intended state. The evolution of the belief distribution can thus be monitored during task to see how quickly and accurately will the agent arrives at the correct hypothesis about the hidden goal.

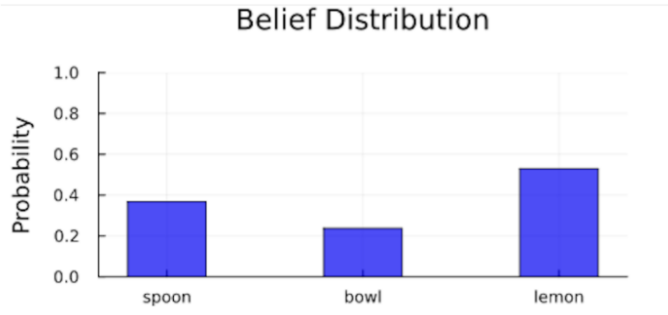

Fig. 1. The belief state of the system after the arm reaches its target object, *lemon*

2) Belief Entropy: Belief entropy measures the uncertainty of A agent belief state. Mathematically, it is the Shannon entropy of the distribution bt . The discrete belief of the system is defined as:

$$H(b_t) = -\sum_{s \in \mathcal{S}} b_t(s) \log b_t(s)$$

This formula calculates the average information content of the belief distribution. When the probability mass is smeared across many states, entropy becomes high and the robot gets uncertain of its state and when the probability mass is concentrated to a single state, the entropy becomes zero and the robot becomes certain of its current state. In our context $H(bt)$ directly measures the uncertainty of robot's internal belief at time t. This metric is computed at each time step in the simulation by plugging the current belief vector into the entropy formula. In our experiment, the entropy typically decreases over time only when human intervention occur. There is an observable spike in the entropy of robot. A prolonged high entropy indicates confusion and ambiguous inputs.

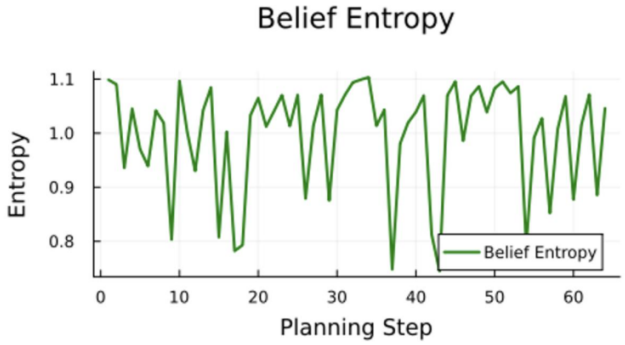

Fig. 2. The belief entropy of state with increasing planning steps

3) Cross Entropy: While belief entropy measures the internal uncertainty of belief, cross-entropy measures the divergence of an agent's belief and the true state. The cross entropy between the true distribution (q) and the estimated distribution (bt) is defined as:

$$\text{CE}(q, b_t) = -\sum_{s \in \mathcal{S}} q(s) \log_2 b_t(s)$$

This cross entropy tells us whether the uncertainty that we got is correct or not. It is simply the minus log of probability of a particular state. Cross entropy directly penalizes the agent for putting low probability on the true state. If the probability in a state is 100 percent, cross entropy is zero and is perfect. When the robot is in its true state and the probability is zero then the cross entropy tends to reach infinity, which means that the robot is hundred percent sure that the current suggested state is wrong. Averaging this metric over time gives an indication of the overall belief accuracy of agent.

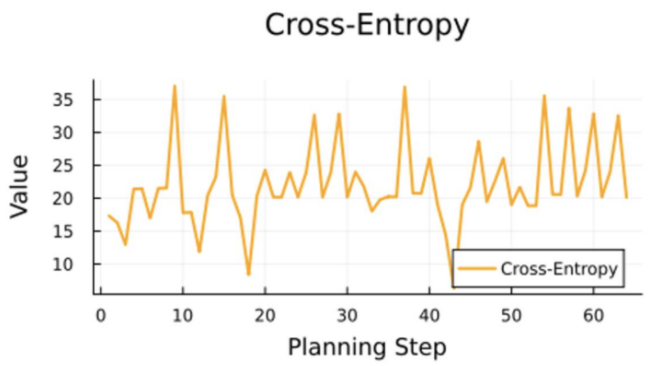

Fig. 3. The cross-entropy value of the system

IV. Results

In a shared autonomy system, a robot collaborates with a human operator to achieve goals in an environment with multiple possible objectives. In this section, we explore the hyperparameter tuning of the Single action controller i.e reward values for state exploration, temperature for belief updater, POMDP solver iterations and MCTS Tree depth value. The following hyperparameters were used in the configuration of the POMDP experiment. These values define the behavior and constraints of the model, ensuring consistency and reproducibility.

TABLE I
Hyperparameter Settings

Hyperparameter	**Value**
Discount Factor (γ)	0.7
Maximum Depth	100
Goal Threshold	2 cm
Default Initial Goal	Bowl
Number of Objects	3
Temperature	0.01

In this experiment, a robot or AI agent is set up to make decisions in a partially observable environment, like navigating to a bowl among three objects. The settings include a discount factor of 0.7, which makes the agent prioritize short-term rewards over distant ones, and a planning limit of 100 steps to keep decisions manageable. The agent considers a task complete if it gets within 2 cm of the bowl, its default target. A temperature of 0.01 ensures the agent mostly sticks to the best-known actions rather than exploring randomly, while the three objects define the complexity of the space it navigates. These hyper-parameters are picked based on experimentation directly and might not be optimal for all cases.

V. Conclusion

This project successfully demonstrates the integration of POMDPs into a robotic task under uncertainty and demonstrated the challenge of human intent inference. By using MuJoCo physics simulator and Sawyer arm using Julia based POMDP framework in this simulator, we created a unique environment for evaluating how probabilistic reasoning enables robot to act when the goal state is not fully observable.

Our approach revolved about modelling human intended goals, represented through a sequence of directional input as a hidden state variable in POMDP. The robot used belief distribution to track its evolving hypothesis, Bayesian updates to include user input and Monte Carlo tree search based planner to select actions to reach to goal. The performance of the system was evaluated using belief entropy and cross-entropy, offering quantitative insights into the robot's confidence and correctness over time.

The experiment showed that even with sparse input, the robot can understand intent with increasing certainty, and reaching the desired goal. The use of belief based reasoning enabled good handling of uncertainty. Overall our work validates POMDPs as an effective approach for human- robot collaboration when state space is not fully observable.

VI. Future Works

We have the following things planned for the future which we believe would help us improve our system further:

1) So far we have tested with just the Monte Carlo Path Planning approach. Testing with value iteration models would certainly be interesting and we believe that the task is also suitable to frame in terms of point-based value iteration planning approach.
2) Evaluate belief updates using human input (consistent or legibility-based) and trajectory similarity to enhance human-robot interaction.
3) Increase the complexity of the environment by increasing the number of objects and varying the environment setups. We would like to incorporate objects which are close to each other see how the robotic agent updates its belief state.

References

[Hsiao et al.(2007)Hsiao, Kaelbling, and Lozano-Perez] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Grasping pomdps. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4685–4692, 2007. doi: 10.1109/ROBOT.2007.364201.

[Katt et al.(2018)Katt, Oliehoek, and Amato] Sammie Katt, Frans A. Oliehoek, and Christopher Amato. Learning in pomdps with monte carlo tree search, 2018. URL `https://arxiv.org/abs/1806.05631`.

[Lauri et al.(2023)Lauri, Hsu, and Pajarinen] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, February 2023. ISSN 1941-0468. doi: 10.1109/tro.2022.3200138. URL `http://dx.doi.org/10.1109/TRO.2022.3200138`.

[Wiering and M. van Otterlo(2012)] M. A. Wiering and editors M. van Otterlo. Reinforcement Learning: State of the Art, Springer Verlag, 2012.

[Xiao et al.(2019)Xiao, Katt, Pas, Chen, and Amato] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. Online planning for target object search in clutter under partial observability. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8241–8247, 2019. doi: 10.1109/ICRA.2019.8793494.