Utilizing Human Feedback in POMDP Execution and Specification

Janine Hoelscher[1,*], Dorothea Koert[1,*], Jan Peters[1,2] and Joni Pajarinen[1]

***Abstract*— In many environments, robots have to handle partial observations, occlusions, and uncertainty. In this kind of setting, a partially observable Markov decision process (POMDP) is the method of choice for planning actions. However, especially in the presence of non-expert users, there are still open challenges preventing mass deployment of POMDPs in human environments. To this end, we present a novel approach that addresses both incorporating user objectives during task specification and asking humans for specific information during task execution; allowing for mutual information exchange. In POMDPs, the standard way of using a reward function to specify the task is challenging for experts and even more demanding for non-experts. We present a new POMDP algorithm that maximizes the probability of task success defined in the form of intuitive logic sentences. Moreover, we introduce the use of targeted queries in the POMDP model, through which the robot can request specific information. In contrast, most previous approaches rely on asking for full state information which can be cumbersome for users. Compared to previous approaches our approach is applicable to large state spaces. We evaluate the approach in a box stacking task both in simulations and experiments with a 7-DOF KUKA LWR arm. The experimental results confirm that asking targeted questions improves task performance significantly and that the robot successfully maximizes the probability of task success while fulfilling user-defined task objectives.**

I. Introduction

Robot support is well-established in industrial applications, where the environment is precisely monitored and fully observable. In contrast, in everyday environments, robots have to handle much more diversified operating conditions including partial observability and unknown object properties.

A partially observable Markov decision process (POMDP) [12] is a common model for task planning which takes uncertainty in both robot actions and observations into account. However, designing a POMDP for a new robotic task is not straightforward and the computational complexity has been limiting the application of POMDPs.

In human environments, users should have the opportunity to get involved in robotic task planning, as they may have preferences regarding the outcome. In addition, humans are likely to perceive or possess additional information about the environment that is inaccessible to the robot's sensors [28].

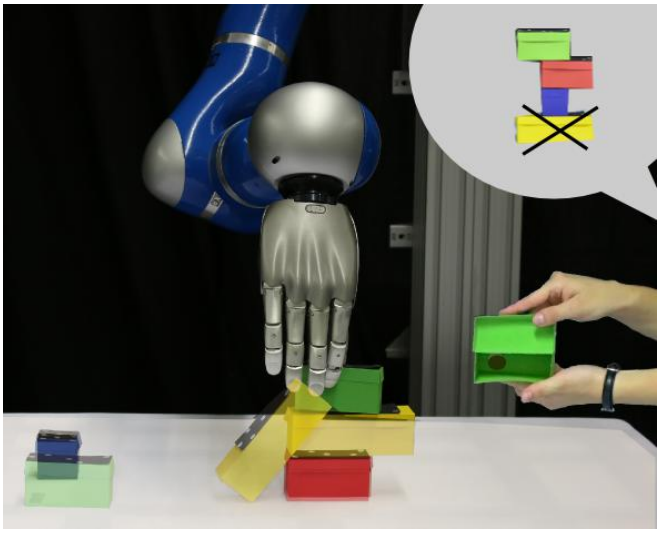

Fig. 1. Utilizing human feedback can be beneficial in tasks where a robot has to deal with partial observations and uncertainty. In addition, users should be able to communicate their objectives on the task outcome to the robot. Our approach addresses both challenges: We propose a novel POMDP algorithm that allows for intuitive user-defined objectives. Moreover, we introduce targeted queries that enable the robot to ask for specific sub-state information such as the weight location inside a specific box.

Therefore, an approach that incorporates both human preferences during task specification and human input during task execution is desirable.

Our main research question is: *How can we leverage human expertise efficiently for planning under uncertainty in both task specification and execution?* Our approach to solve this question is of twofold nature to account for both, an intuitive way for non-expert users to define task objectives and the ability for the robot to ask for human knowledge in uncertain situations.

First, our approach provides humans with an intuitive way to define their goals for the task. As non-expert users cannot be assumed to be familiar with technical details of POMDPs, we introduce a system for objective definitions based on object properties. Therefore, we define a set of logic sentences users may choose from through a graphical user interface. To solve the resulting offline planning problem, we introduce Logical Particle Based Policy Graph Improvement (LPPGI), a novel algorithm which is based on the policy graph improvement (PGI) algorithm [20]. This algorithm maximizes the expected probability of satisfying logic objectives beyond the traditional total sum of rewards objective.

Second, our approach enables the robot to ask a human for specific information during task execution. To this end, we incorporate targeted queries in a POMDP, which allow

*both first authors contributed equally

This work was supported by EU Horizon 2020 project RoMaNS and ERC StG SKILLS4ROBOTS, project references #645582 and #640554, and, by German Research Foundation project PA 3179/1-1 (ROBOLEAP)

[1]Intelligent Autonomous Systems, TU Darmstadt, Germany
[2]MPI for Intelligent Systems, Tuebingen
jhoelsch @ cs.unc.edu,
{koert,peters,pajarinen} @ ias.tu-darmstadt.de

the robot to inquire about objects' properties in the current world state. The robot may select from different questions in order to receive the piece of information that is most relevant for completing the task.

We demonstrate our approach in a box stacking task in both simulation and experiments with a 7 DoF robotic arm, as illustrated in Figure 1. In particular, we show that this task can be simplified by allowing the robot to ask targeted queries and we demonstrate how non-experts can specify different objectives through a user interface.

II. Preliminaries & Related Work

In this section, we briefly recapitulate the formal definition of POMDPs as well as solving approaches and remaining challenges. In subsection II-B, we focus on related methods for interaction between human users and (robotic) agents in the POMDP context.

A. Partially Observable Markov Decision Processes

Markov Decision Processes (MDPs) [2] provide a principled way of modeling an agent interacting with its surrounding environment. POMDPs (Partially Observable MDPs) are an extension to MDPs that take uncertainty in state observations into account. A POMDP is defined as a tuple $\langle S, A, O, \Gamma, \Omega, R, b_0 \rangle$ [12] of states S, actions A, observations O, and the following:

- Transition probability $\Gamma = P(s'|s,a) : S \times A \to S$ denoting the probability for a transition to state s' after performing action a in state s,
- Observation probability $\Omega = P(o|s',a) : S \times A \to O$ denoting the probability of observing o after executing action a and ending in the next state s',
- Reward function $R(s,a) : S \times A \to \mathbb{R}$ specifying the immediate reward at state s when choosing action a.

The goal is to compute a policy π^* which maximizes the total expected discounted reward

$$\pi^* = \arg\max_{\pi} \quad \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)|\pi\right] \tag{1}$$

where π denotes the policy, t the time step, T the optimization horizon, and $0 < \gamma \leq 1$ is the discount factor. Moreover, a POMDP has an initial belief distribution $b_0(s)$ that models the agent's initial knowledge about the world state. The belief $b(s)$ is updated with new observations at every time step t. In POMDPs, the belief is a sufficient statistic for optimal decision making and can be computed from past actions and observations. More details regarding the belief update can be found, for example, in [25].

Many different algorithms for POMDP optimization have been proposed, both exact [27], [30] and approximate [25], [26], [31], [19]. However, large POMDPs remain hard to solve, as finite-horizon planning problems are PSPACE-complete [21], and infinite-horizon POMDPs are even undecidable [16]. For more details on POMDPs, readers are referred to [12], [25].

B. Related Work

POMDPs have been used for planning under uncertain conditions in many different robotic contexts, notably in human environments [29], [10], [22]. However, when humans are a common part of the environment, new requirements and opportunities for POMDPs arise. Users desire to adapt tasks according to their preferences [29], [13], [9] and may provide crucial information to robotic agents [3], [8], [1]. In the following, we present existing strategies for interaction between users and robots, while focusing on improving task performance by querying the user and ways of customizing the task goal. Furthermore, we discuss computational methods for solving POMDPs.

One strategy for the robot to improve its planning performance is to ask a human user for information. This approach is well established e.g. in reinforcement learning [5]. In a POMDP model, information can be included by updating the reward function [3], the observation function [4], or the transition model [8]. Yet, these methods require a large amount of human feedback. Instead, new information can also be directly incorporated into the current belief state. Armstrong et al. [1] introduce an additional action a_{oracle} asking about the current state of the system, which is used to reset the belief state. This belief update can also take uncertainty about the oracle's correctness into account [23].

However, providing complete state information can be cumbersome or even infeasible for human users. While it is relatively straightforward to extend a POMDP model by adding query actions targeting specific state properties, optimizing the resulting POMDP policy is challenging. Asking targeted questions requires a number of information gathering actions, resulting in a large action space. We show that an appropriate POMDP algorithm can plan targeted queries in practice, and that targeted queries can significantly increase performance in a robotic application while minimizing user effort.

A second direction for improving interaction between users and robots is to let human users express their objectives for a robotic task in a natural way. To learn user intentions, [29], [13], [9], [17] gather human goal specifications from feedback during task execution. However, these methods require a user to be attentive and to interfere several times during task execution. The direct approach of letting the user define the reward function requires potentially untrained users to define and balance rewards and costs for different states. A more natural way for users to specify task objectives is through logic sentences describing desired state properties, while the robot maximizes the probability of satisfying those objectives. Existing logic based POMDP formulations try to maximize the reward [24] or satisfy logic sentences fully [6]. Instead, our approach assumes a probabilistic Markovian dynamics and observation model but aims to maximize the probability of satisfying all logic sentences. [7], [14] maximize the probability of a POMDP system to always remain in a desired part of the state space.

We investigate a more general problem where a user can

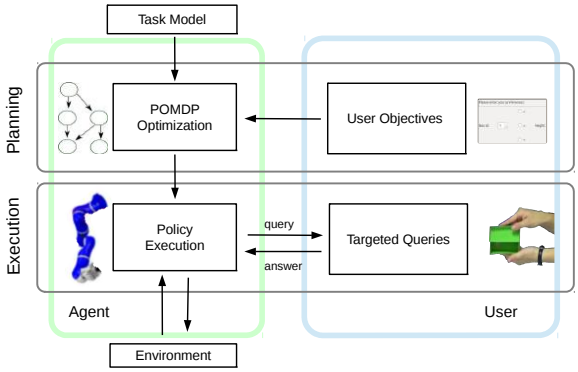

Fig. 2. Our new approach on interactive planning under uncertainty enables user-friendly objective definition using LPPGI for offline planning, and information retrieval during task execution through targeted queries answered by the user.

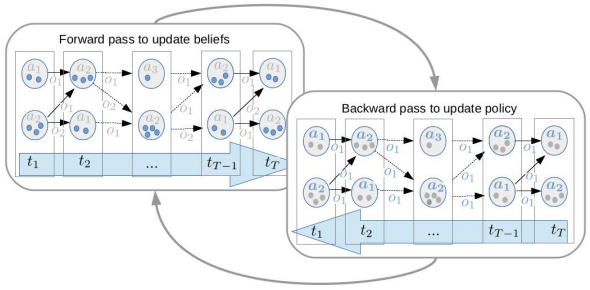

Fig. 3. In a policy graph, each node (gray) defines an action and edges represent particular observations that lead to the next node. Similar to Particle based Policy Graph Improvement (PPGI), Logical PPGI (LPPGI) improves the policy graph in two phases: During the forward pass, the algorithm samples a subsequent state s' and an observation o for each state particle (blue) at each time step following the current policy. In the backward pass, the algorithm optimizes for each policy graph node a policy that consists of the best action and of the best next node for each observation. While PPGI maximizes the expected reward, LPPGI maximizes the probability of satisfying predefined preferences.

specify any kind of task requirements for any subset of time steps. We propose a new algorithm based on *Policy Graph Improvement* (PGI) [20] and the Particle based PGI [19] (PPGI) algorithm. PGI [20] optimizes POMDPs through iterative improvement of a fixed-size policy graph, allowing for linear scaling of the optimization time. PPGI [19] also scales to large state spaces [18], which are common in robotic tasks. In Section III, we introduce a novel algorithm based on PPGI that maximizes the probability of logic sentence success in large state spaces. In contrast, earlier work related to maximizing the success probability [7], [14] relies on algorithms working only in small to moderate size state spaces.

III. Interactive Planning under Uncertainty

In this section, we present a novel approach for utilizing human input during both specification and execution of robotic tasks under uncertainty. We introduce a new algorithm, Logical Particle based Policy Graph Improvement (LPPGI), and discuss how it allows for the definition of user objectives in the form of logical sentences, referred to as *preferences* in the following. In addition, we introduce *targeted queries* enabling the robot to request information about sub-state properties from a user. We then define a POMDP model applicable to robot manipulation tasks. Figure 2 depicts both parts of the proposed approach.

A. Logical Particle Based Policy Graph Improvement

In this section, we present the novel Logical Particle based Policy Graph Improvement (LPPGI) algorithm. Similar to PGI [20], LPPGI iteratively improves a policy graph of fixed size but in contrast to PGI, it computes a policy that maximizes the expected probability of fulfilling predefined preferences, given as a boolean function $C_t(s, a)$.

Most POMDP solvers compute policies by optimizing the expected reward, as defined in (1). In contrast, LPPGI searches for a policy π^* that optimizes the expected probability of fulfilling user preferences

$$\pi^* = \arg\max_{\pi} \quad \mathbb{E}\left[\prod_{t=0}^{T-1} C_t(s_t, a_t)|\pi\right], \tag{2}$$

where T denotes the planning horizon and the user preferences are encoded by a predefined boolean function $C_t(s, a)$. We call the optimization problem in (2) a maximum success problem. It is related to maximizing the probability of the system remaining in a safe part of the state space [7], [14] for the duration of the complete task, but, in the maximum success problem, $C_t(s, a)$ can be time step specific.

In the maximum success problem, the sufficient statistic can be represented as an improper probability distribution, that is, $\hat{b}_t(s)$ is the probability that the system is in state s given previous observations and given that all previous constraints are satisfied [14]. In more detail, we define $\hat{b}_0(s) = b_0(s)$, $\tilde{b}_t(s) = \hat{b}_t(s)/||\hat{b}_t||_1$, and

$$\hat{b}_{t+1}(s'|\hat{b}, a, o) = \eta_o P(o|s', a) \sum_s C_t(s, a) P(s'|s, a)\tilde{b}(s), \tag{3}$$

where $||\hat{b}_t||_1$ is at each time step t the expected probability of having satisfied $C_k(s, a)$ in previous time steps $k < t$, $\eta_o = 1/\sum_{s,s'} P(o|s', a)P(s'|s, a)\tilde{b}(s)$ is a normalizing constant, $\hat{b}(s_0)$ denotes the initial belief, and $\hat{b}_{t+1}(s'|b, a, o)$ denotes the updated belief for the next state s'. When setting $V_T(\hat{b}) = ||\hat{b}||_1$, that is, setting the next to last time step value function to the probability of satisfying all constraints, the value function V follows from Bellmann's optimality principle as

$$V_t(\hat{b}) = \max_a \left[\sum_{o,s,s'} P(o, s'|s, a)\tilde{b}(s)V_{t+1}(\hat{b}_{t+1}(\cdot|\hat{b}, a, o))\right]. \tag{4}$$

Our derivations for (3) and (4) are similar to those in [14], except that in [14] the derivations are intended for more

general hybrid controls and are assuming an observation space identical to the state space.

Now, in order to compute policies for complex real world problems, we derive a policy graph algorithm as depicted in Figure 3. Similarly to the PGI algorithm, we use a fixed size policy graph with T layers of nodes q. At layer t, at policy graph node q the policy executes the action a according to the transition probability $P_t(a|q)$ and transitions to the next layer node q' depending on the made observation o according to $P_t(q'|q,o)$. $P_t(a|q)$ and $P_t(q'|q,o)$ are deterministic. The value function $V_t(s,q)$ depends on the policy graph node instead of the belief. The algorithm iterates between forward and backward passes. During the forward pass, we simulate the current policy and update the belief according to Eq. (3)

$$\hat{b}_{t+1,q'}(s') = \sum_{q,o,a,s} C_t(s,a) P(o,s'|s,a) P_t(a|q) P_t(q'|q,o) \hat{b}_{t,q}(s),$$

where $\hat{b}_{t,q}(s)$ denotes the belief at time step t in node q. Note that we do not need normalization η_o due to summing over all observations. In the backward pass, we update the policy by maximizing Eq. (4) as follows.

For the next to last time step T we set $V_T(s,q) = 1$ and recursively compute for each graph node q and each observation o at time step t the best next layer graph node

$$q_{t+1}(a,o,q) = \arg\max_{q'} \sum_{s,s'} \big(C_t(s,a) \hat{b}_{t,q}(s) \\ P(o,s'|s,a) V_{t+1}(s',q') \big).$$

The best action $a^*_{t,q}$ is then computed as

$$a^*_{t,q} = \arg\max_{a} \sum_{s,o,s'} \big(C_t(s,a) \hat{b}_{t,q}(s) \\ P(o,s'|s,a) V_{t+1}(s', q_{t+1}(a,o,q)) \big).$$

Next, we compute the updated value function

$$V_t(s,q) = \sum_{a,s',o,q'} C_t(s,a) P(s',o|s,a) P_t(q',a|q,o) V_{t+1}(s',q'),$$

where in $P_t(q',a|q,o) = P_t(a|q) P_t(q'|q,o)$ we set

$$P_t(a = a^*_{t,q}) = 1, \qquad P_t(q' = q_{t+1}(a^*_{t,q}, o, q)|q,o) = 1.$$

Figure 3 provides an overview of the forward and the backward pass. To scale to large state spaces, similarly to PPGI, we use particle filtering in LPPGI to approximate probability distributions. Algorithm 1 provides the pseudo code for LPPGI.

B. LPPGI for User Friendly Objective Definition

For non-expert human users, the design of complex reward functions in partially observable tasks is not trivial. To enable a more intuitive way of interaction during task specification we propose to use LPPGI to incorporated user-defined preferences, in form of object specific logic sentences, in a POMDP model. Formally, we define the m-th logic sentence part $c_m(s)$ as

$$c_m(s) := \Big[\forall \text{ obj}_n \in s : p_i(\text{obj}_n) = \alpha \implies p_j(\text{obj}_n) \ \Delta \ \beta \Big]$$

with $\Delta \in \{>, \geq, <, , \leq, =, \neq\}$. Here, the first part, $\forall$ obj$_n \in s$: p_i(obj$_n$) $= \alpha$, of the definition selects objects with

Algorithm 1: LPPGI

π = `LPPGI(`$b_0(s)$, π_0`)`
  while No convergence and time limit not exceeded **do**
    b =`ParticleForwardPass(`$b_0(s)$, π`)`
    π =`ParticleBackPass(`b`)`
  end

π =`ParticleBackPass(`b`)`
  for Time step $t = T-1$ to 0 **do**
    foreach Policy graph node q_t at layer t **do**
      foreach Action a **do**
        $V_{a,o,q'} = 0$,$V_{a,o} = 0$ for all o, q'
        for $i = 1$ to N **do**
          $s_i \sim b_{q_t}(s)$,$s'_i \sim P(s'|s_i,a)$,
          $o_i \sim P(o|s'_i, a)$
          $C_a = C_t(s_i, a)$
          foreach Next node q' **do**
            $V_{a,o_i,q'} = V_{a,o_i,q'} + C_a$
              `Sim(`s'_i,t+1,q',π`)`
          end
        end
        foreach Observation o **do**
          $q_{t+1}(a,o,q_t) = \text{argmax}_{q'} V_{a,o,q'}$
          $V_{a,o} = V_{a,o,q_{t+1}(a,o,q_t)}$
        end
        $V_a = \left(\sum_o V_{a,o}\right)/N$
      end
      $a_{t,q} = \text{argmax}_a V_a$
    end
  end

V =`Sim(`s_i, t, q, π`)`
  $V = C_t(s_i, a_{t,q})$
  for Time step t to $T-1$ **do**
    $s'_i \sim P(s'|s_i, a_{t,q})$, $o_i \sim P(o|s'_i, a_{t,q})$
    $q' = q_{t+1}(a_{t,q}, o_i, q)$, $V = VC_{t+1}(s', a_{t+1,q'})$
    $q = q'$, $s_i = s'_i$
  end

b =`ParticleForwardPass(`$b_0(s)$, π`)`
  $b_{0,0}(s) = b_0(s)$
  for Time step $t = 0$ to $T-1$ **do**
    Set $b_{t+1,q'}(s')$ to an empty set for all q'
    foreach Policy graph node q at layer t **do**
      for $i = 1$ to $N_{t,q}$ **do**
        $s_i \sim b_{t,q}(s)$
        if $C_t(s_i, a_{t,q})$ **then**
          $s'_i \sim P(s'|s_i, a_{t,q})$, $o_i \sim P(o|s'_i, a)$
          $q' = q_{t+1}(a_{t,q}, o_i, q)$
          Add state s'_i to $b_{t+1,q'}(s')$
        end
      end
    end
  end

the i-th property equal to α whereas the second part, $p_j(\text{obj}_n)\ \Delta\ \beta$, defines a constraint on the j-th property of the selected objects. Similarly, it is possible to express preferences that only have to be fulfilled by only one object, such as "at least one blue box must be positioned above height 3", i.e.,

$$c_m(s) := \Big[\exists\ \text{obj}_n \in s : p_i(\text{obj}_n) = \alpha \wedge p_j(\text{obj}_n)\Delta\ \beta\Big].$$

As prior work on web query interfaces [11] indicates that users employ logical conjunctions more than disjunctions or negations, we define $C_t(s_t, a_t)$ as a conjunction of these logic sentence parts. The resulting boolean function for the LPPGI algorithm as introduced in Section III-A can then be computed as

$$C_t(s_t, a_t) := \begin{cases} 1 & \text{if } t \in [\tau_{\text{min,m}}, \ldots, \tau_{\text{max,m}}] \implies c_m(s_t)\ \forall m \\ 0 & \text{else} \end{cases}$$

where $\tau_{\text{min, m}}$ and $\tau_{\text{max, m}}$ characterize the time interval during the task for which logic sentence part $c_m(s)$ has to be fulfilled. The part can either be continuously active or only at certain times. A logic sentence solely holding for terminal states is a special case of this formulation.

In a discounted reward POMDP, the discount factor emphasizes short term rewards and induces a cost on postponing actions. However, selecting a good discount factor often requires deeper expertise. For the user-friendly objective definition, we allow the user to set a hard limit on the total number of time steps instead. As the maximum number of time steps in a trial is an intuitively understandable property, non-expert users should be capable of selecting meaningful values.

C. Targeted Queries

Some object properties in real world robotic tasks may be easily accessible to a human user but not to the robot's sensors. This could be the case if the user has prior knowledge or if the robot's sensory modalities are limited. In our approach for interactive planning under uncertainty, we, therefore, provide the robot with the opportunity to ask questions. Previous approaches define an oracle action [1], [23], asking for the complete current world state. In contrast, our approach only queries specific states of a specific object. Hereby, we allow the robot to request crucial information in risky situations and minimize the effort for human users. We define several targeted query actions a_{query} alongside other POMDP actions. Hereby, the robot can ask for specific information, as we define targeted queries as actions $a_{\text{query}} =< obj_n, p_j >$, i.e., the robot may select one property p_j of one specific object obj_n for its inquiry. For now, we assume the user's answer to always be correct, however, including uncertainty on this answer is also possible with our chosen POMDP approach. Moreover, a hard limit on the number of queries can be selected. If this limit is reached, the agent can not query the human user anymore but must rely on its own information gathering actions for additional information gain.

D. POMDP Model for Object Manipulation

In this section, we introduce the POMDP model that is used in the experimental evaluations. In general, this model can be applied to various contexts of manipulating multiple objects sequentially. Similarly to the POMDP model in [19], our model defines an object by its properties: $obj := p_1 \times \cdots \times p_J$. A state s consists of N objects: $s = \{obj_1, \ldots, obj_N\}$ while $p_i(obj_n)$ denotes the property i of object n.

The model can be used in connection with rewards or in connection with user-defined preferences as introduced in Section III-B.

In case of rewards, rewards are only collected in goal states G for which the manipulation task is completed:

$$R(s, a) = \begin{cases} 1 & \text{if } s \in G \\ 0 & \text{else} \end{cases}.$$

There is no further support or punishment for other states. Thereby, we avoid reward balancing as untrained users might not be able to select meaningful reward values for different states. We also define failure states: If such a state is reached, the trial is over, and there is no reward.

IV. Experimental Evaluation

In this section, we present an experimental evaluation of the proposed approach. As an example application, we choose a box stacking task with potential human interaction and partial observability due to unknown weight distribution in the boxes. This example represents robot manipulation tasks involving objects with potentially unknown properties, such as assembly or household tasks. In the following, we define the box stacking task and then present our experimental results for policy planning.

A. Object Manipulation Under Uncertainty: Box Stacking

In this task, a robot builds a tower out of a given set of paper boxes. Due to hidden weights, the centers of mass of the boxes are unobservable for the robot. The weights are realized by a coin as depicted in Figure 5(b). The robot can only determine a box's weight position by placing the box on top of the tower and by observing whether or not it stays on top. Such a robot movement is shown in Figure 4. Alternatively, the robot may use targeted queries as introduced in Section III-C to ask about the weight of a specific box. We assume that humans have prior knowledge of the weight positions since they can open the boxes.

The boxes in our setup have different fixed properties including length l, height h, and position of the hidden weight, w. A state consists of all available boxes b_i, $S = \{b_1, \ldots b_B\}$, and each of the boxes is defined by its properties, i.e. $b_i = \langle l, h, w, x, z\rangle$, where x denotes the horizontal position of the box and where z designates the height from the table. For example, with 5 boxes this results in a large state space of $933,120$ states (combination of box- and weight positions).

Agent actions are defined as $a =< i, x >$, specifying a box i and its new position x on top of the tower w.r.t. the box

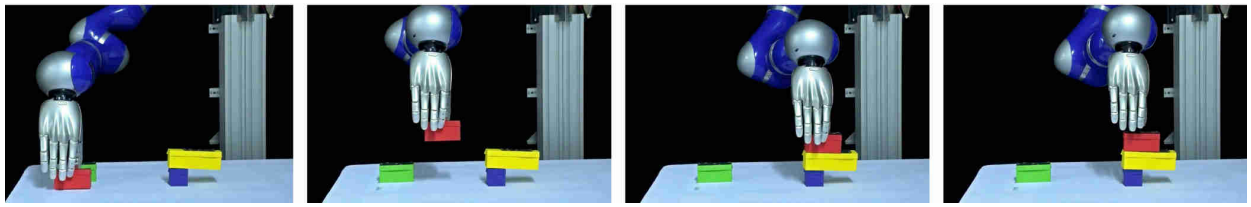

Fig. 4. We demonstrate our approach on a 7 DoF robot arm. A planned action performed by the robot consists of several steps, i.e., the robot grasps the selected box, moves it towards the tower, positions it according to the action, and releases the box.

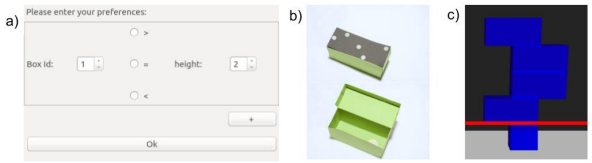

Fig. 5. (a) Users can define objectives in form of logical sentences through a graphical user interface. (b) The boxes' centers of mass are unobservable for the robot due to hidden weight locations in form of a coin inside the box. (c) Box stacking simulation with a limited maximum falling height (red). The trial is over if a box falls from higher than this limit.

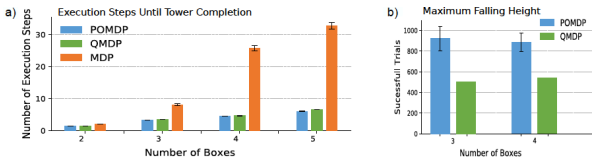

Fig. 6. (a) We compare MDP, QMDP (with value iteration) and POMDP (with PPGI) on the box stacking task. Both the MDP and the QMDP take significantly more actions because they do not account for partial observability or do not take information gathering into account, respectively. (b) The difference becomes even more obvious when a maximum falling height of 1 is introduced and the QMDP's success rate drops.

underneath (we discretize position values to keep the size of the action state space feasible). Additionally, the robot may move boxes back to their origin. At each time step the robot may also ask a targeted query $a_{\text{query}} =< i, weight >$ about the weight position of box i. Tower stability is determined by calculating the position of the center of mass on each level of the tower. Boxes that fall down are assumed to be moved back to their initial position. Observations express the number of boxes toppling during or after the execution of an action. All transitions and observations are assumed to be deterministic. If rewards are used, a reward is only received in a terminal goal state if the tower structure contains all boxes. In addition, a trial is over if a box falls from higher than a fixed height limit. This limitation motivates information gathering on a lower height level before stacking boxes high. Experiments start with a single box of size 1 located in the middle of the designated construction area as the tower foundation. The initial belief b_0 is a uniform distribution over possible weight configurations for the system of boxes. We provide a graphical user interface, as shown in Figure 5(a), for user preference definition.

B. Experiments and Results

We perform three series of experiments with offline POMDP planning. First, we evaluate whether POMDPs are appropriate for such a manipulation task. Then, we analyze in which situations querying a human simplifies the task. Additionally, we demonstrate how to solve the task when using LPPGI for objective definitions on a real robot.

1) Stacking Without a Human: In the first set of experiments, we investigate if our manipulation task benefits from a POMDP model in comparison to an MDP or QMDP model. For details on the QMDP algorithm the interested reader is referred to [15]. We compute 15 POMDP policies by applying the PPGI algorithm with a constant policy length and width of 15. For evaluation, we simulate each policy 1000 times with state particles randomly sampled from a prior belief. In the QMDP and MDP case, policies are computed deterministically via value iteration. A discount factor of 0.9 serves as an incentive for limiting the number of execution steps. The simulation confirms that the POMDP planning process works well for up to five unknown boxes in a reasonable amount of time (on average 40 sec for 5 boxes). Repeated executions of the planning task lead to almost identical policies admitting a low standard error. The MDP does not take partial observations into account during execution and therefore highly depends on the initial state assumption. That is why we allow the MDP to start again with a different initial state assumption after the optimal number of actions was executed. However, Figure 6(a) shows that the MDP requires significantly more actions for successful tower completion since it is unable to model the belief during execution. QMDPs do account for partial observability and therefore achieve better results. However, unlike POMDPs, QMDPs do not gather information, and therefore are outperformed as well. The difference becomes obvious if a maximum falling height of 1, as sketched in Figure 5(c), is introduced and memorizing weight positions gains importance. Figure 6(b) illustrates that in this case the POMDP has a significantly higher success rate than the QMDP. Overall, these results indicate that the POMDP model is well suited and necessary for the tower stacking task.

2) Stacking With a Human Answering Questions: In the following experiments, we evaluate the effect of targeted queries and compare them to oracle actions [1]. Experiments contain a fixed number of 5 boxes and the maximum falling height varies.

As illustrated in Figure 7(a), targeted query actions, as well

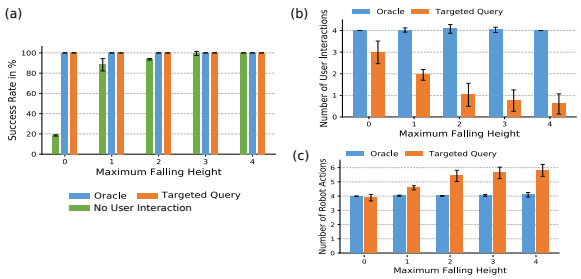

Fig. 7. (a) The opportunity to request information from the user in form of oracle actions or targeted queries significantly increases the success rate, in particular for lower maximum falling heights. (b) While oracle actions always provide full state information and therefore require user queries for all boxes, targeted query actions adapt the number of user interactions according to the maximum falling height. (c) In the case of targeted queries the robot balances between its own box stacking actions for information gathering and information retrieval from the human. This leads to a slight increase of robot actions compared to the oracle actions.

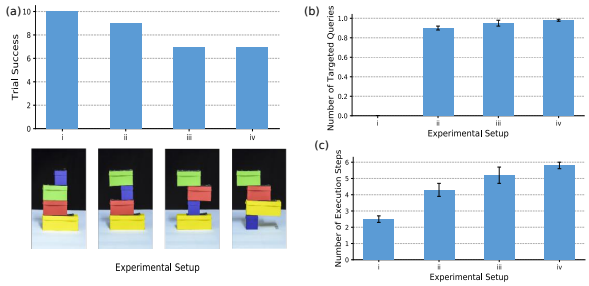

Fig. 8. (a) Through user objective definition, the blue box is requested to be at four different positions in the completed tower structure, resulting in four experiments. The share of successful trials decreases as balancing an increasing number of boxes on top of the smallest one becomes more and more demanding. (b) The algorithm adapts the use of targeted queries to the task planning difficulty. (c) The execution steps increase as more information gathering actions become necessary for lower tower locations of the blue box.

as oracle actions, exhibit a significant increase in task rate, in particular in combination with a low maximum falling height. This is expected since in these cases the center of mass of each box has to be determined before the actual stacking can start. In settings with higher falling limits, it is less expensive to determine a weight position with exploratory actions as those potentially lead to tower growth. It is desirable for task performance that only necessary questions are being asked. While the oracle action always provides the full state information, targeted queries only ask for one specific box's properties. As depicted in Figure 7(b), the number of used targeted queries adapts to different maximum falling heights. The planning algorithm balances well between information gain and task progress, rendering the process significantly less cumbersome for users. As can be seen in Figure 7(c), this leads to a slightly higher number of robot actions being performed in experiments with targeted queries due to exploratory robot actions.

In addition, the experiments show that planning does not require significantly more effort, even though the size of the action space is increased in case of targeted queries.

3) Robot Experiments: While the presented experiments focused on the evaluation of single components in simulation, we now demonstrate targeted queries and user preference planning on a real robot. The robot hardware consists of a 7 DoF Kuka LWR robot arm and a DLR Hit Hand II as well as an Optitrack system with 6 cameras tracking the boxes and providing observations. These components and the POMDP planner are connected through ROS. We employ four paper boxes of different sizes as shown in Figure 8(a). The maximum falling height is set to 1 and provide the opportunity for one targeted query action. In addition, the position of the blue box in the tower is defined through user preferences to be at different specific heights. For each predefined position of the blue box, we repeated the experiment 10 times. The robot successfully optimizes plans for building towers, and fulfills all user preferences. As shown in Figure 8, the robot adapts its use of targeted queries to the task difficulty (more information is required if the smallest box has to be placed at a lower position). Even though construction is not always successful due to material shortcomings and tracking noise, in the majority of trials the robot successfully builds a tower according to user-defined preferences and by incorporating information from targeted queries.

V. Conclusions

In this work, we propose a new POMDP based approach for robotic tasks in complex uncertain environments involving interaction with a non-expert human user. Our contributions are of two-fold nature: First, we show that requesting specific information about the task using targeted queries is beneficial for task success. Second, we define logic sentences through which users can express preferences about the manipulation task's outcome prior to task planning. To this end, we introduce a novel algorithm maximizing the probability of satisfying these logic sentences.

Empirical evaluations of a box stacking task, in simulation and with a real robot, demonstrate that our approach succeeds in determining task plans that fulfill the user's specifications. Additionally, the experimental results show that targeted queries can improve the robot's performance significantly compared to self-sufficient information gathering. Furthermore, user effort can be minimized if only partial state information is required in queries.

While in this work LPPGI was solely used for offline planning, future applications may benefit from online replanning, for example, based on user query input. We plan future user studies with non-experts to further evaluate the suitability of the proposed user-friendly objective definitions for task specification as well as to evaluate how human subjects respond to different levels of robot queries. In addition, user input can be prone to uncertainty. Incorporating this uncertainty into the POMDP model is another line of possible future work. Furthermore, we consider testing our model on a broader variety of manipulation and assembly tasks.

REFERENCES

[1] Nicholas Armstrong-Crews and Manuela Veloso. Oracular partially observable Markov decision processes: A very special case. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, pages 2477–2482. IEEE, 2007.

[2] Karl J Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[3] Amin Atrash and Joelle Pineau. A bayesian reinforcement learning approach for customizing human-robot interfaces. In *Proc. of the 14th International Conference on Intelligent User Interfaces*, pages 355–360. ACM, 2009.

[4] Amin Atrash and Joelle Pineau. A bayesian method for learning POMDP observation parameters for robot interaction management systems. In *Workshop at the International Conference on Automated Planning and Scheduling (ICAPS)*, 2010.

[5] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[6] Krishnendu Chatterjee, Martin Chmelík, Raghav Gupta, and Ayush Kanodia. Qualitative analysis of POMDPs with temporal logic specifications for robotics applications. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 325–330. IEEE, 2015.

[7] Jerry Ding, Alessandro Abate, and Claire Tomlin. Optimal control of partially observable discrete time stochastic hybrid systems for safety specifications. In *Proc. of the American Control Conference (ACC)*, pages 6231–6236. IEEE, 2013.

[8] Finale Doshi-Velez, Joelle Pineau, and Nicholas Roy. Reinforcement learning with limited reinforcement: Using bayes risk for active learning in POMDPs. *Artificial Intelligence*, 187:115–132, 2012.

[9] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1879–1884. IJCAI, 2007.

[10] Jesse Hoey, Axel Von Bertoldi, Pascal Poupart, and Alex Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *Proc. of the International Conference on Vision Systems*, volume 65, page 66, 2007.

[11] Steve Jones, Shona McInnes, and Mark S Staveley. A graphical user interface for Boolean query specification. *International Journal on Digital Libraries*, 2(2):207–223, 1999.

[12] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[13] Abir-Beatrice Karami, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Partially observable Markov decision process for managing robot collaboration with human. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 518–521. IEEE, 2009.

[14] Kendra Lesser and Meeko Oishi. Reachability for partially observable discrete time stochastic hybrid systems. *Automatica*, 50(8):1989–1998, 2014.

[15] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings*, pages 362–370. Elsevier, 1995.

[16] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proc. of the Innovative Applications of Artificial Intelligence Conference (IAAI)*, pages 541–548. AAAI, 1999.

[17] Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 294–302. ACM, 2017.

[18] Joni Pajarinen and Ville Kyrki. Robotic manipulation in object composition space. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6. IEEE, 2014.

[19] Joni Pajarinen and Ville Kyrki. Robotic manipulation of multiple objects as a POMDP. *Artificial Intelligence*, 247:213–228, 2015.

[20] Joni K Pajarinen and Jaakko Peltonen. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in Neural Information Processing Systems*, pages 2636–2644. NIPS, 2011.

[21] Christos H Papadimitriou and John N Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[22] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3):271–281, 2003.

[23] Stephanie Rosenthal, Manuela M Veloso, and Anind K Dey. Learning accuracy and availability of humans who help mobile robots. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1501–1506. AAAI, 2011.

[24] Scott Sanner and Kristian Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1140–1146. AAAI, 2010.

[25] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, pages 1–51, 2013.

[26] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172. NIPS, 2010.

[27] Edward J Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

[28] Alexander Sorokin, Dmitry Berenson, Siddhartha S Srinivasa, and Martial Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2117–2122. IEEE, 2010.

[29] Tarek Taha, Jaime Valls Miró, and Gamini Dissanayake. POMDP-based long-term user intention prediction for wheelchair navigation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3920–3925. IEEE, 2008.

[30] Erwin Walraven and Matthijs TJ Spaan. Accelerated vector pruning for optimal POMDP solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3672–3678. AAAI, 2017.

[31] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017.