Artificial Intelligence 247 (2017) 213–228

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# Robotic manipulation of multiple objects as a POMDP

Joni Pajarinen*, Ville Kyrki

*Department of Electrical Engineering and Automation, Aalto University, Finland*

ARTICLE INFO

*Article history:*
Received in revised form 15 February 2015
Accepted 1 April 2015
Available online 3 April 2015

*Keywords:*
POMDP
Planning under uncertainty
Task planning
Manipulation
Unknown objects
Multiple objects
Cluttered environment

ABSTRACT

This paper investigates manipulation of multiple unknown objects in a crowded environment. Because of incomplete knowledge due to unknown objects and occlusions in visual observations, object observations are imperfect and action success is uncertain, making planning challenging. We model the problem as a partially observable Markov decision process (POMDP), which allows a general reward based optimization objective and takes uncertainty in temporal evolution and partial observations into account. In addition to occlusion dependent observation and action success probabilities, our POMDP model also automatically adapts object specific action success probabilities. To cope with the changing system dynamics and performance constraints, we present a new online POMDP method based on particle filtering that produces compact policies. The approach is validated both in simulation and in physical experiments in a scenario of moving dirty dishes into a dishwasher. The results indicate that: 1) a greedy heuristic manipulation approach is not sufficient, multi-object manipulation requires multi-step POMDP planning, and 2) on-line planning is beneficial since it allows the adaptation of the system dynamics model based on actual experience.

## 1. Introduction

For a service robot, physical interaction with its environment is an essential capability. As the application areas of service robotics are extending to complex unstructured environments, robotic manipulation has become an important focus area within robotics research.

In complex environments, the robot's knowledge about its environment is incomplete and uncertain. To operate in such environments, robots can employ different mechanisms. First, a robot may use on-line sensing to attempt to gain more information about the environment. Second, sensory measurements can be used directly in a feedback control loop to adapt to small disturbances. Third, the uncertainty can be taken into account on the level of planning the actions.

Planning under uncertainty with imperfect sensing can be modeled as a partially observable Markov decision process (POMDP). While POMDPs have been one of the actively pursued research directions within AI, they have not been applied very widely in robotic manipulation. This is partly because the manipulation planning problems have intrinsic characteristics such as continuous state spaces which make the application of POMDP solvers less straightforward, and partly because the manipulation planning approaches have only recently advanced to the point where the explicit modeling of uncertainty becomes tractable. The question which robotic manipulation problems benefit from the explicit modeling of uncertainty in a POMDP framework remains open to a large extent.

* Corresponding author.
*E-mail addresses:* Joni.Pajarinen@aalto.fi (J. Pajarinen), Ville.Kyrki@aalto.fi (V. Kyrki).

http://dx.doi.org/10.1016/j.artint.2015.04.001

In this paper, we consider a multi-object manipulation planning problem with the environment state estimated by imperfect sensors. The dynamics of the system are considered to be partly unknown, which supports the goal of long term autonomy of the robotic system. Our high-level research question is in which situations explicit planning under uncertainty is beneficial. The manipulation planning problem is considered on task level, that is, the desired result of planning is the best action to be performed. While motion planning for an individual action, such as grasping an object, is out-of-the-scope for this paper, the relationship of such an individual action to observations of the overall system state and the world model dynamics are modeled and learned during operation.

The theoretical contributions of the paper are two-fold: first, a POMDP model for multi-object manipulation is proposed. As a particular novel contribution, the model considers the effect of visual occlusions on observations and success of actions. Moreover, the dynamics of the world model, in particular related to the success of actions, are updated during operation as more information is gained. Second, we propose a new POMDP method which is applicable to manipulation planning. In particular, it does not require a discrete world model but instead samples the world model to construct policies. The method is also scalable, does not require heuristics, can handle uncertainty in the world model, and allows online planning, which is important when the world model is not accurate. Furthermore, the method produces compact policies in a predefined time. This can be beneficial in a robotic setting where easy to inspect policies may give new insights into the problem.

The proposed approaches are experimentally evaluated both in simulation and with a real robot. The experiments demonstrate that multi-step, longer horizon planning is beneficial in complex environments with clutter. In particular, POMDPs are beneficial if a particular problem has some of the following characteristics: 1) the problem requires weighting the value of information gathering versus collecting immediate rewards such as lifting objects to get a better view on other objects, 2) the world model is uncertain and thus it should be updated, for example when some objects are harder to grasp than others, or 3) the sequence of actions matters such as when objects occlude each other even partially. Altogether, the paper is the first to propose long term POMDP planning for manipulating many objects in a high dimensional, unknown, and cluttered environment.

## 2. Related work

### 2.1. Partially observable Markov decision processes

A partially observable Markov decision process (POMDP) [1] defines the optimal policy for a sequential decision making problem while taking into account uncertain state transitions and partially observable states. This makes POMDPs applicable to diverse application domains such as robotics [2], elder care [3], tiger conservation [4], and wireless networking [5]. However, versatility comes with a price, the computational complexity of finite-horizon POMDPs is PSPACE-complete [6] in the worst case.

Because of the high computational complexity, state-of-the-art POMDP methods [7–11] use different kinds of approximations. There are at least two causes for the intractability of POMDPs: 1) state space size, and 2) policy size. State-of-the-art POMDP methods yield good policies even for POMDP problems with hundreds of thousands of states [7,8] by trying to limit policy search to state space parts that are reachable and relevant for finding good policies. However, in complex real-world problems the state space can be still much larger. In POMDPs with discrete variables, the state space size grows exponentially w.r.t the number of state variables. In order to make POMDPs with large state spaces tractable, there are a few approaches: compressing probability distributions into a lower dimension [12], using factored probability distributions [13,14], or using particle filtering to represent probability distributions [9,10]. Particle filtering is particularly attractive, because an explicit probability model of the problem is not needed. In fact, in order to cope with a complex state space, we use particle filtering in the online POMDP method presented in more detail in Section 3.2.

Assuming the problem of state space size solved, the problem of policy size still remains. In the worst case, the size of a POMDP policy grows exponentially with the planning horizon. Offline POMDP methods [11] take advantage of the piecewise linear convexity (PWLC) of the POMDP value function to keep the size of the policy reasonable. Some offline POMDP methods use fixed size policies. A common approach is to use a fixed size (stochastic) finite state controller [15,16] as a policy. The monotonic policy graph improvement method [17] utilizes a fixed size *policy graph*, an idea which is also adopted here.

Contrary to offline approaches, online POMDP methods [18] compute a new policy at each time step. Online planning starts from the current belief and can thus concentrate on only the part of the search space that is currently reachable. Moreover, restarting planning from the current belief allows the online planner to correct planning “mistakes”, which an inaccurate world model caused in earlier time steps. This is especially relevant in robotic manipulation in service settings where an accurate world model is difficult to estimate for example due to unknown objects. Online POMDP methods usually represent the policy as a policy tree. Techniques such as pruning can be used in order to reduce the size of the policy tree, but this does not solve the problem of exponential growth of the policy tree w.r.t. the planning horizon. The online POMCP method of Silver et al. [9] uses particle filtering to address the state space size problem and Monte-Carlo tree search to explore the policy space. However, for best results, POMCP requires a problem specific heuristic [9]. POMCP is also not designed to produce compact policies. The online POMDP approach that we propose allows for long planning horizons by using a compact, fixed-size policy graph [17]. Because of the compact policy size, the policy can be inspected by a domain expert.

It is useful to notice the relationship of on-line POMDP solvers to *receding horizon control* (RHC) [19,20], a popular feedback control technique. With RHC, a control problem is formulated as optimization over a fixed-length time horizon and the solution to the optimization provides a plan. At each time step the optimization problem is solved anew and the first step of the resulting plan is applied. The idea of fixed-length receding horizon has been applied also to POMDPs, e.g. in [13]. However, there is one significant difference between RHC and POMDP solvers: In RHC the controls are typically optimized directly. In contrast, our approach as a POMDP solver optimizes a feedback policy for the moving horizon.

In manipulation, and more generally in robotics, the world model is often uncertain and thus it is necessary to learn the world model during online operation. The goal then is to maximize total reward while taking into account that the current world model is not accurate and that actions can yield information about the world model. For a discrete POMDP, a natural Bayesian approach is to model transition and observation probabilities with Dirichlet distributions [21–23]. Following this, our probability model uses Beta distributions to model uncertainty in object specific grasp probabilities. Few papers [24,25] exist on using a POMDP model with uncertain probabilities in robotics. Ross et al. [24] apply an online POMDP approach to simulated robot navigation with Gaussian distributions with unknown parameters. Bai et al. [25] present an offline POMDP planning approach for robot motion planning with unknown model parameters and validate their approach in simulation. However, we are not aware of prior robotic manipulation research that uses a POMDP model with uncertain probabilities.

### *2.2. Manipulation under uncertainty*

Manipulation planning under uncertainty is not a new problem to be considered in robotics. Already in the early 1980's, Lozano-Péres et al. [26] considered the automatic synthesis of fine-motion actions under initial robot pose uncertainty using compliant motion, preimages and backward chaining. The originally sensorless decision-theoretic line of research can be seen to continue to this day with extensions over the years to e.g. grasping [27] and a probabilistic setting [28]. A good summary of the current state in this line of work can be found in [29].

There is a recent trend to integrate task and motion planning, see [30] for an overview. However, these approaches do not address directly the problem of uncertainty. The only exception is the recent work by Kaelbling and Lozano-Péres [31], where preimage backchaining is used for belief-space planning in a hierarchical framework. The approach handles probabilistic uncertainty by using a deterministic approximation of the domain and replanning after each time step. Our work differs from this approach: we consider the interactions of the manipulation actions, that often occur in multi-object manipulation, as well as update the world model based on on-line experience.

There is a tradition to formulate robot navigation problems as POMDPs, for an overview see [32], or a recent study for long time horizon POMDP planning [33]. In manipulation, the use of the formulation is not common. Hsiao et al. [34] proposed the partitioning of the configuration space of grasping with one uncertain degree of freedom to yield a discrete POMDP which can be solved for an optimal policy. In grasp planning, the state-of-the-art includes probabilistic approaches with a short time horizon. The goal can be formulated either as positioning the robot accurately as in [35] or maximizing the probability of a successful grasp as in [36,37]. The short-term planning can also be extended to include information gathering actions [38]. In contrast to the above, this paper considers manipulation of multiple objects which are unknown and where the sequence of actions has a significant effect.

Recent work by Dogar and Srinivasa [39] proposes manipulation of multiple objects using grasping and pushing primitives. The approach uses pushing to collapse the uncertainties of the object locations as well as to clear clutter in the scene. The planning is performed at the level of object poses. Monso et al. [40] proposed to formulate clothes separation as a POMDP. In contrast to our work, the approach of Monso et al. is environment specific. Monso et al. rely on a clothes separation specific state space definition, which models the number of clothes in each area. We model object attributes, associated probabilities, and grasp probabilities, in any kind of environment.

## 3. Multi-object manipulation: a POMDP

In multi-object manipulation, a robot performs actions on several objects. In particular, the robot may grasp objects, move them, or use them in another way to accomplish some predefined goals. In this paper, we focus on the problem of deciding how to manipulate unknown objects in a crowded environment. Because the environment is crowded, only parts of the objects can be observed by visual sensors. In addition to uncertain observations, real-world manipulation problems have uncertain action consequences, especially when the robot does not have a model of the objects beforehand, or when the robot does not observe the objects well. For example grasping or moving an object may fail, because the shape or location of the object differs from the observed one. Real-world problems often have several (possibly conflicting) goals. As a practical example consider putting dirty dishes from a table full of dishes into a dishwasher: the goal is to maximize the number of dirty dishes in the dishwasher, minimize the number of clean dishes in the dishwasher, and minimize the execution time. In order to address the issue of complex objectives, uncertain observations, and uncertain action effects, this paper models the problem of manipulating multiple unknown objects as a partially observable Markov decision process (POMDP).

Planning of manipulation, for instance grasp planning, is traditionally considered as a geometrical problem. However, in unstructured environments with unknown objects the current state-of-the-art approaches often plan individual actions (e.g., a grasp) directly based on the observed environment [41]. We follow the same idea so that the planning is performed on the level of semantic actions and locations, while the execution of individual actions is then performed based on the currently observed scene. However, our approach also models the interplay of the immediate sensor measurements to both observation and system models. For example, the rate of visual occlusion modulates the probability of correct observations and successful completion of actions.

We begin by defining a POMDP, then describe a new online POMDP planning method which is suitable for complex problems such as multi-object manipulation, and finally describe how to model multi-object manipulation in crowded environments as a POMDP with an application of moving dirty dishes into a dishwasher.

### *3.1. What is a POMDP?*

A POMDP is a model that defines optimal behavior for a given Markovian problem, taking into account uncertainty in observations as well as action effects over a potentially long time horizon. In a POMDP, a set of hidden Markov models, one for each action choice, describes the temporal dynamics of the problem and the optimization objective is defined by assigning a reward to each action in each possible situation. In a specific application, rewards should reflect real value, e.g. monetary cost.

Formally a POMDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, O, b_0 \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $\mathcal{O}$ is the set of observations. The state set includes all possible states of the world, in which the agent is assumed to operate in. $P(s'|s,a)$ is the transition probability to move from state $s$ to the next time step state $s'$, when action $a$ is executed. $R(s,a)$ yields the real-valued reward for executing action $a$ in state $s$ and $O$ denotes the observation probabilities $P(o|s',a)$, where $o$ is the observation made by the agent, when action $a$ was executed and the world moved to the state $s'$. Lastly, $b_0(s)$ is the initial state probability distribution, also known as the initial *belief*. In a finite-horizon POMDP, the goal is to optimize the expected reward

$$E\left[\sum_{t=0}^{T-1} R(s(t), a(t))|\pi\right], \tag{1}$$

where $T$ is the horizon, $s(t)$ is the state, and $a(t)$ the action chosen at time step $t$ by the policy $\pi$.

Because the states are not fully observable, the current state cannot be used for decision making as in fully observed models. Instead, the *belief* $b(s)$, a probability distribution over world states, is maintained to make (optimal) decisions at each time step. Starting from the initial belief $b_0(s)$, the belief is updated at each time step. After performing action $a$ and observing $o$ the updated belief $b' = b'(s'|b,a,o)$ can be obtained from the current belief $b = b(s)$ using the Bayes formula $b'(s'|b,a,o) = \frac{P(o|s',a)}{C} \sum_s P(s'|s,a)b(s)$, where $C$ is a normalizing constant.

To give an intuitive idea of how POMDPs can be applied in practice, we will now give short examples for the transition and observation probabilities, and the reward function. In a POMDP, the transition probability $P(s'|s,a)$ models the uncertainty in action effects: what is the probability to move a cup successfully from a table (part of state $s$) into a dishwasher (part of $s'$), when the action $a$ is "move cup into dishwasher"? The observation probability $P(o|s',a)$ models the uncertainty in observations: what is the probability of observing a cup as dirty (observation $o$), when it is dirty (part of state $s'$) and we are executing action $a$ "look at cup"? Finally, the reward $R(s,a)$ explicitly specifies the optimization goal: gain positive reward for moving (action $a$) a dirty cup (part of state $s$) into the dishwasher.

### *3.2. Online policy graph POMDP using monotonic value improvement*

In this paper, as often in robotic applications, the state space of the robotic manipulation task is high dimensional. The state space has exponential size in the number of discrete state variables, and includes uncertain grasp success probability distributions. Because of the complex state space, POMDP methods based on exact probability representations are not applicable. We present a new online POMDP method based on the monotonic policy value improvement algorithm [17] proposed by us earlier. The next subsection briefly introduces the method from [17] followed by the extensions: 1) the new method uses particle filtering to represent probability distributions and estimate values in a way that takes advantage of the policy graph (Section 3.2.2), instead of using a discrete tabular probability distribution representation [17], and 2) the new method is transformed from an offline [17] method into an online method in a policy graph specific way (Section 3.2.3).

#### *3.2.1. POMDP policy and method*

We represent the policy of the agent (the robot) as a policy graph $G$ (see Fig. 1 for an example policy graph) beginning from time step $t = 0$ and ending at the planning horizon $t = T - 1$. The policy graph consists of layers of graph nodes, one layer for each time step. Each graph node defines a conditional plan for the robot to follow: which action to perform, and depending on the observation made, to which next layer node to transition next.

The policy improvement approach in [17] uses dynamic programming to improve each policy graph layer at a time. First, the approach computes the belief $b_t(s,q)$ at each layer $t$ and each graph node $q$ starting from the initial belief $b_0(s)$ in the first layer. Then, starting from the last layer and moving one layer at a time towards the first layer, the approach computes for each node in the layer a new policy (action, observation edges), which maximizes the expected reward for the belief at the current node. The expected reward is computed from the immediate reward and the next layer value function

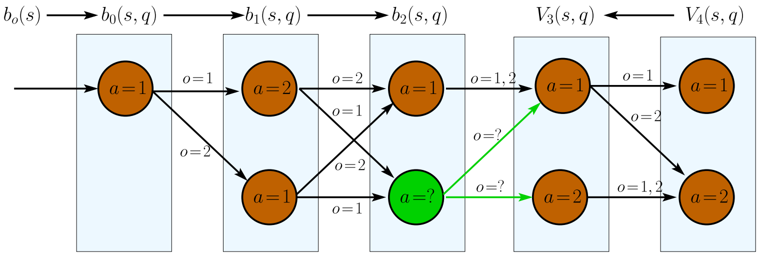


**Fig. 1.** Illustration of a policy graph node update in the monotonic policy graph value improvement algorithm for POMDPs.

$V_{t+1}(s,q)$, which yields the expected reward when starting from state $s$ and graph node $q$ in layer $t+1$ and following the policy graph until layer $T-1$. After optimizing the policy for a policy graph node: If the new policy at the node is identical to the policy of another already optimized node in the same layer, we sample a new belief and re-optimize the policy at the node for the sampled new belief. Because having multiple graph nodes at the same layer with the same policy does not improve the value of the policy graph, re-optimizing such "redundant nodes" does not decrease the value of the policy graph. The whole policy graph improvement procedure guarantees monotonic improvement of policy value. For algorithmic details, see [17].

In order to keep computations tractable, we use a policy graph with fixed width and depth. This circumvents the problem of exponential growth of a search tree, allows for manual inspection of a compact policy, and enables us to convert the offline approach to an online one.

### *3.2.2. Particle filtering*

The method in [17] assumes a discrete "flat" POMDP. In order to deal with a large state space, we use particle filtering to approximate beliefs and for estimating values.

**Belief representation and update.** We represent a belief $b(s)$, a probability distribution over $s$, as a finite set of particles [32], that is, a weighted set of state instances $s^j$. The belief is $b(s) = \sum_j w^j \delta(s, s^j)$; $\sum_j w^j = 1$; $0 \leq w^j \leq 1$, where $w^j$ is the particle weight and $\delta(s, s^j) = 1$ when $s = s^j$ and zero otherwise. What a state actually is depends on the application: Section 3.3 defines a state for multi-object manipulation.

We use two kinds of belief updates. The first one is the commonly used update of the current belief $b(s)$, when an action has been executed and an observation made. This belief update is used in initializing the policy graph and for sampling new beliefs for redundant policy graph nodes (please, see Section 3.2.1 on redundant nodes). The second kind of belief update projects the belief $b_t(s,q)$ over world states and policy graph nodes to the next layer belief $b_{t+1}(s,q)$, using the current policy. The second belief update is used in each improvement round. We will next discuss the belief updates in more detail.

In the first belief update, the action $a(t)$ and observation $o(t+1)$ are given. We sample a next time step state $s^j(t+1)$ for each current state $s^j(t)$ according to the application specific dynamics (state transition) model $P(s^j(t+1)|s^j(t), a(t))$. We then compute the new particle weight $w^j(t+1)$ as the product of the old weight and the observation probability: $w^j(t+1) = w^j(t)P(o(t+1)|s^j(t+1), a(t))$. As usual, to prevent particle impoverishment, we resample particles, when the effective sample size drops below a threshold (0.1 in the experiments).

In the second belief update, for updating the belief $b_t(s,q)$, a particle consists of a weight $w^j(t)$ and a state/node pair $(s^j(t), q^j(t))$. To sample a new particle $(s^j(t+1), q^j(t+1))$ using an $(s^j(t), q^j(t))$ pair, we first get the action $a(t)$ for node $q^j(t)$. Then, we sample a new state $s^j(t+1)$ from $P(s^j(t+1)|, a(t))$. Next, we sample an observation $o(t+1)$ from $P(o(t+1)|s^j(t+1), a(t))$. Finally, the observation edge for observation $o(t+1)$ of the graph node $q^j(t)$ yields the new graph node $q^j(t+1)$. In this update, the particle weights do not change.

As a side remark, note that our approach differs from existing particle filtering based approaches. In order to improve the policy, we use the current policy for finding a belief distribution over graph nodes, but other state-of-the-art POMDP methods based on particle filtering [9,10] select an action and observation to find a new belief for which to compute a policy. In other words, other POMDP methods use a constant amount of particles to represent a single belief, but we use a constant amount of particles to represent the belief over a policy graph layer (a time step) and each graph node is assigned particles proportional to the probability of the graph node.

**Value estimation.** In order to determine the best action and observation edges for a policy graph node, the method has to estimate the value for each action–observation-next node triplet. From these triplets the method can then select for each action the highest value observation-next node pairs and based on these select the highest value action. To do this efficiently, we follow Algorithm 1 in [10]. The algorithm samples state transitions and observations for each action and for the sampled observation simulates the value for each next controller node. Bai et al. [10] represent the policy as a possibly cyclic finite state controller, but we use instead an acyclic policy graph. However, no significant modifications are necessary because the algorithm is based on simulation. Furthermore, the bound for the approximation error induced by sampling, shown in Theorem 1 in [10], also applies here: the error is bounded by a term that decreases at the rate of $\mathcal{O}(1/\sqrt{N})$, where $N$ is the number of samples.

In the implementation we do not actually sample states from a belief, but just go through all particles, one at a time, and utilize the particle weight for value estimation. When particles have identical weights, going through particles is more efficient than sampling them. Particles often have identical weights during policy improvement.

**Complexity.** In one improvement round, the method improves the policy at each policy graph node by going through policy graph layers from the last layer to the first. To improve the policy at a policy graph node the method uses the value estimation technique described above. Value estimation simulates state trajectories from a policy graph node until the end of the policy graph taking linear time w.r.t. the planning horizon. That is, the method improves a linear number of policy graph layers and at each policy graph layer propagates state samples for a linear number of times w.r.t. the planning horizon. Therefore, the worst case complexity of one policy improvement round of the POMDP method is quadratic w.r.t. the planning horizon. In the experiments in Section 4, the method performed well. In the future, one could parallelize the algorithm to utilize multiple CPU cores (easily because of the particle representation of probabilities), or use a fixed sampling depth.

### 3.2.3. *From offline to online*

Because of the computational and modeling restrictions discussed previously, we transform the offline POMDP method into an online one. Similarly to the receding horizon control (RHC) approach [20,42] in automatic control we re-plan at each time step up to a finite horizon. Intuitively, we use a moving window that at each time step shifts one step to the right over the policy graph (imagine this with the help of the policy graph in Fig. 1), discards the first layer, and adds a new layer at the end. At the beginning, the agent optimizes the policy graph for several improvement rounds for the initial belief. Then in following time steps the agent estimates the new belief and constructs a new policy for the belief, as follows: 1) initialize the new policy graph with the layers $2, \ldots, T-1$ of the previous policy graph; 2) add a new last layer to the policy graph with random actions, and add random observation edges to the layer preceeding the last layer; 3) use the regular policy graph improvement method on the new policy graph. The basic idea here is to initialize the current policy graph using the policy graph of the previous time step, and then optimize the policy graph for the current belief. Because of the initialization, the required number of improvement rounds during online operation is then less when compared to offline optimization.

## 3.3. *Multi-object manipulation as a POMDP*

We discuss now a general POMDP framework for modeling multi-object manipulation. Later, in Section 3.3.1, we then show how the POMDP framework can be applied to the problem of moving dirty dishes into a dishwasher.

In multi-object manipulation, the robot has to decide at each time instance which object to manipulate. We consider problems, where the world consists of $N$ objects with varying attributes. The total number of actions is $\sum_i |A_i|$, where $|A_i|$ denotes the number of possible actions for object $i$. In each time step, the action of the robot changes the spatial locations and poses of the objects, and the robot makes an observation about the changed state of the world. Our POMDP model uses discrete actions and observations. However, instead of forcing the robotic planning problem into a manageable discrete state space as is done e.g. in [40], we use a POMDP method based on particle filtering (discussed in Section 3.2) that allows us to maintain complex object information required for efficient multi-object manipulation.

**State space and actions.** The state space consists of semantic object locations (e.g. “on table”, “in a dishwasher”), object attributes, and historical data of observations and action successes for each object. The model assumes that the semantic location of an object is constant over time unless a manipulation action successfully changes it. However, because an online planning approach is used, the planning always restarts from the current belief taking into account the most recent measurements.

Formally, the POMDP state $s = (s_1, s_2, \ldots, s_N)$ is a combination of object states $s_i = (s_i^{\text{loc}}, s_i^{\text{attr}}, s_i^{\text{hist}})$ where $s_i^{\text{loc}}$ is the semantic object location, $s_i^{\text{attr}}$ the object attributes, and $s_i^{\text{hist}}$ compressed historical information of action successes and object attribute observations. The action success information consists of a count of succeeded $n_i^{\text{succ}}$ and failed $n_i^{\text{fail}}$ grasps for each object. Because of the finite number of objects the number of action counts is finite. Similarly, as discussed in more detail below, the number of different object attribute observations is finite. Therefore, $s_i^{\text{hist}}$ has finite dimensionality, and the POMDP state can be stored and operated on efficiently. Note that the POMDP states have the Markov property because the probability for the next state depends only on the current state (and action).

The observation history contains information of past observations of object attributes. Past object attribute observations can be used to compute the probability distribution over an object's attributes. Additionally, these are needed during planning because future observations of the attributes cannot be assumed statistically independent, because the main source of observation uncertainty is occlusion. In contrast, unless the occlusion changes, we assume that an identical observation of the attribute is made (note that we assume differently occluded observations independent). We assume that the probability of making the correct observation depends on how occluded the object is (we discuss this in more detail shortly). In more detail, $s_i^{\text{hist}}$ contains the observation made in each occlusion setting. For example, in the experiments objects can be temporarily lifted: in addition to the current occlusion setting, we store the observation for each object which was temporarily lifted and which is otherwise in front of the observed object. Note that because of the finite number of objects the number of occlusion settings is finite, and thus the observation history has finite size.

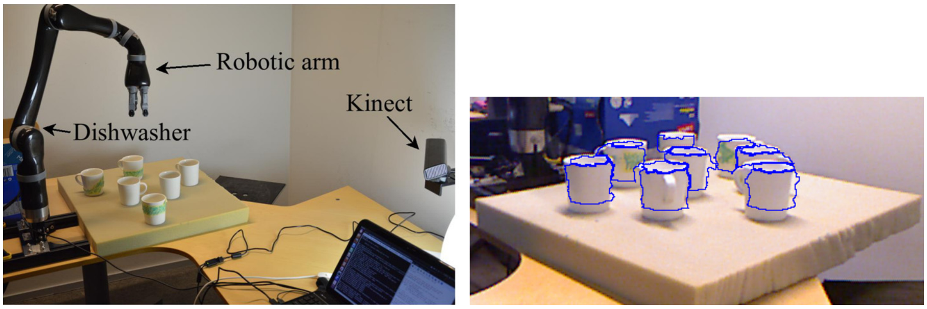


**Fig. 2.** Experimental setup. **Left:** A Kinova Jaco robotic arm manipulates objects placed on the table. A Microsoft XBOX Kinect acts as a monocular visual sensor for capturing RGB-D point clouds. In the experiments, the goal is to pick up dirty objects, here cups marked with a green color, from the table and place them into the "dishwasher", represented by the blue box on the far left. **Right:** An image captured by the Kinect sensor. Object edges are depicted in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Occlusion ratio.** The action success probability and the observation probability of an object depend on how occluded the object is. Because we do not have models for the objects, the occlusion is modeled using a model free *occlusion ratio*. The reasoning is that the higher the occlusion, the smaller the probability of success in actions or observations. In the experiments, we capture a point cloud, segment the point cloud into objects, compute edges for all objects using 2-D information, and then find out how much the edges of objects touch each other. The right hand side figure in Fig. 2 shows edges found for segmented objects in a scene. When the edge of object A, which is closer to the visual sensor, touches the edge of object B, object A occludes object B.

Consider computing the occlusion ratio for object B. Denote with TOT the perimeter of the 2D contour of object B, that is, the total number of 2D pixels for which the number of neighboring 2D pixels, which are part of object B, is less than eight. Denote with TOU the touching edge between A and B, that is, the number of 2D pixels in B which have atleast one neighboring 2D pixel in object A (when B is occluded by several objects, just use the 2D pixels of the occluding objects). The occlusion ratio for object B is 1, when TOT subtracted by TOU is smaller than TOU, 0 when TOU $= 0$, and otherwise TOU/(TOT $-$ TOU). The reasoning is that when an object almost completely occludes another object, TOT is roughly double TOU. Thus an occlusion ratio of 1 corresponds to totally occluded and an occlusion ratio of 0 to no occlusion at all. The convenience variable $s_i^{\text{occl}}$ denotes the occlusion ratio of object $i$.

In this paper, POMDP state transitions are based on sampling. When an object A is sampled to be moved, so that it does not occlude another object B anymore, it is straightforward to update the occlusion ratio of B by removing the touching edge between A and B. However, if there is an object C, which occludes A (edges of A and C touch), but not B, and A is moved away, then there is a possibility that C could occlude B after the removal of A. We call this *occlusion inheritance*. For simplicity, we do not take occlusion inheritance into account in the experiments and leave it as future work.

**Grasp probability.** We assume that occlusion affects the grasp probability of all objects in a similar way, but, in addition, we assume that each object has unknown properties that affect the grasp probability of that specific object: we do not know beforehand what kind of grasp properties each object has. For example a cup that has fallen down may be harder to grasp, than another cup, which is standing upright (see Fig. 8b for an example). The probability of a successful grasp is modeled as

$$\begin{aligned} &P(\text{grasp succeeded}|s_i^{\text{occl}}, s_i^{\text{hist}}) = E[p_i^{\text{succ}}] \\ &\quad p_i^{\text{succ}} \sim Beta(p_i^{\text{succ prior}} n^{\text{prior}} + n_i^{\text{succ}}, (1 - p_i^{\text{succ prior}}) n^{\text{prior}} + n_i^{\text{fail}}), \end{aligned} \tag{2}$$

where $p_i^{\text{succ prior}}$ is the occlusion ratio specific grasp success prior probability and $n^{\text{prior}}$ is the strength of the prior. In the experiments, we mapped the occlusion ratio to the grasp success prior probability $p_i^{\text{succ prior}}$ using a simple exponential function

$$p_i^{\text{succ prior}} = \exp(-\theta_{\text{G}1} s_i^{\text{occl}} + \theta_{\text{G}2}), \tag{3}$$

where $\theta_{\text{G}1}$ and $\theta_{\text{G}2}$ are parameters that can be experimentally estimated from object grasps, for example, using two different occlusion ratios.

Note that we model the grasp probability as the mean of the Beta distributed random variable $p_i^{\text{succ}}$. It would be possible to use a more complex model during planning, in which one would sample the grasp probability from the Beta distribution, but we expect this would increase the number of particles needed for planning.

**Observations.** We assume that the semantic locations and dependencies (which cup is in front of which cup) are fully observed and that grasp success is also fully observed. At each time step the agent observes whether the grasp succeeded and makes an observation about object attributes. Using these observations, we can compute a probability distribution over object attributes, which is needed for sampling the initial POMDP belief and for displaying attribute probabilities. Note that if grasp success or semantic locations are not fully observed, then we cannot estimate the initial POMDP belief directly using grasp success and object attribute observations. Instead, we could update at each time step an (approximate) belief according to the current action and observation and use that as the initial POMDP belief. However, in many applications, including the dishwasher application further down, semantic locations such as "object on table", "object in dishwasher", and thus also grasp success, are fully observed. Specifically, in the experiments, we use visual sensory input. To determine whether an object has been successfully grasped, we compare the image before and after the grasping action. If we do not detect the object at the same location anymore, then we assume the grasp succeeded. One could also use a tactile sensor for detecting grasp success.

As discussed earlier, we assume that the robot observes an object identically unless the occlusion changes. Denote with $o_i^j$, the observation for object $i$ when in the $j$th occlusion setting, and with $a_i^j$ the action performed when observing $o_i^j$, then the attribute probability given the history is

$$\begin{aligned} & P(s_i^{\text{attr}}|o_i^1,\ldots,o_i^M,a_i^1,\ldots,a_i^M) \\ &= \frac{P(o_i^1,\ldots,o_i^M|s_i^{\text{attr}},a_i^1,\ldots,a_i^M)P(s_i^{\text{attr}}|a_i^1,\ldots,a_i^M)}{P(o_i^1,\ldots,o_i^M|a_i^1,\ldots,a_i^M)} \\ &= P(s_i^{\text{attr}}|a_i^1,\ldots,a_i^M)\prod_{j=1}^{M}P(o_i^j|s_i^{\text{attr}},a_i^j)/\sum_{s_i^{\text{attr}}}\prod_{j=1}^{M}P(o_i^j|s_i^{\text{attr}},a_i^j), \end{aligned} \tag{4}$$

where we assumed that observations are conditionally independent given the object attributes, but if needed and computationally possible one can use joint probabilities. We assume that attributes (e.g. color) do not change over time, and thus actions do not influence object attributes: $P(s_i^{\text{attr}}|a_i^1,\ldots,a_i^M) = P(s_i^{\text{attr}})$. In the experiments, we assumed $P(s_i^{\text{attr}})$ is uniform.

### 3.3.1. *Dirty cups into dishwasher*

We now demonstrate how the framework can be used to model the problem of moving dirty cups from a table into a dishwasher as a POMDP (another realistic application could be moving dishwasher-safe cups, instead of dirty cups, into the dishwasher). In this problem, the robot can gain more information of attributes by removing occlusions and gain information about the object specific grasp probability through successful and failed grasps.

**State space and actions.** In addition to the grasping and observation history discussed in Section 3.3, the world state consists of the semantic location $s_i^{\text{loc}} = \{\text{TABLE, DISHWASHER}\}$, and the attributes $s_i^{\text{attr}}$ of an object include dirtyness $s_i^{\text{dirty}} = \{\text{CLEAN, DIRTY}\}$. The robot can perform three kinds of actions. The *FINISH* action terminates the robot actions and assigns a negative reward to dirty dishes remaining on the table. The *LIFT* action tries to lift an object to expose the objects behind it and allows the agent to gather more information about the occluded objects. A small negative reward representing time cost is associated with the action. Note that the action takes less time than moving the object into the dishwasher. The *WASH* action tries to move an object into the dishwasher (in the experiments, a box). If the move succeeds, the state of the object changes from TABLE to DISHWASHER. If the move succeeds and the moved object is dirty, then a large reward is obtained. If the move succeeds and the object is clean, a large negative reward is obtained. Failed grasps cause a small negative reward accounting for the time cost. Note that when implementing the model, we can compute the reward for the *WASH* action as the expected next time step reward using the grasp success probability, instead of deferring reward computation until the grasp has happened in the next time step.

**Observations.** At each time step the agent observes whether the grasp succeeded and the dirtyness of the $k$ nearest objects (in the experiments $k = 2$) which were occluded by the moved cup. In total, $2^{k+1}$ possible observations. In the experiments, we model the conditional probability of observing cup $i$ as dirty when it is dirty with

$$P(o_i = \text{DIRTY}|s_i^{\text{dirty}} = \text{DIRTY}, s_i^{\text{occl}}) = \exp(-\theta_{\text{D1}}s_i^{\text{occl}} + \theta_{\text{D2}}), \tag{5}$$

where $\theta_{\text{D1}}$ and $\theta_{\text{D2}}$ are parameters that can be, similarly to the grasp probability, experimentally estimated from captured point clouds and object labels. The probability of observing a cup as dirty when it is clean is modeled identically with

$$P(o_i = \text{DIRTY}|s_i^{\text{dirty}} = \text{CLEAN}, s_i^{\text{occl}}) = \exp(-\theta_{\text{C1}}s_i^{\text{occl}} + \theta_{\text{C2}}), \tag{6}$$

where parameters $\theta_{\text{C1}}$ and $\theta_{\text{C2}}$ are also estimated in the same way.

## 4. Experiments

The experiments follow the scenario described above. The scene is observed by an RGB-D sensor (Microsoft Kinect) and a 6-DOF Kinova Jaco arm with an integrated 3-fingered hand is used to manipulate the objects. The objects belong to two classes: clean white cups and cups with green "dirt" representing dirty objects. Fig. 2 illustrates the experimental setup: to the left a picture of the setup, and to the right an image captured by the Kinect sensor.

**Rewards.** The robot receives a reward at each time step. The reward depends on the action executed and the current state of the world. As discussed in Section 3.3.1, the robot can execute three different kinds of actions. The *FINISH* action terminates the problem and accumulates a reward of −5 for each dirty cup on the table. Similarly, to limit experiment run times, after ten time steps, the problem is terminated and a reward of $-5$ for each dirty on the table given. The *LIFT* action lifts an object up and yields a reward of $-0.5$ for both failed and succeeded grasps. The *WASH* action moves an object into the dishwasher. If the move succeeds, then the reward is $+5$ for a dirty object and $-10$ for a clean object. For a failed move the reward is $-0.5$. In our dishwasher application, there was no well determined objective. Rewards were designed based on the researchers' understanding of what the objective could be, for example, for a person employing the robot. In general, rewards should reflect the actual goal [43]. For example, if the robot owner sees equal value in a successfully washed dirty cup and skipping the waiting time on ten robot actions, then the owner could assign a reward of $+5$ to a washed dirty cup and a penalty of $-0.5$ to performing any action.

**Methods.** The *POMDP planning* method described in Section 3.2 is initialized by 10 offline policy improvement rounds. Then, at each time step 4 improvement rounds for the current belief are executed. To evaluate the benefit of planning under uncertainty, the POMDP approach is also compared against heuristic decision making: The *heuristic manipulation* method assumes that observations are accurate and deterministic. It tries to move the dirty cup that has the highest grasp success probability into the dishwasher. If no cup is observed dirty it performs the FINISH action. In the experiments, we used two versions of the heuristic method: one which updates grasp probabilities according to the grasp success history and another which does not remember any grasp history.

**Point cloud into a world model.** In the experiments, the visual sensor captures a point cloud, from which we extract objects, their color, and information on how they occlude each other. From these we estimate grasp and observation probabilities and use these probabilities to plan which action to perform. In more detail, first the Kinect sensor captures an RGB-D point cloud of the visual scene. Without using prior information we segment[1] the point cloud into objects. From the 2D-image, we determine the edge of each object and how much it touches other objects' edges (see edges in the right hand side image of Fig. 2). Using the object edges, we compute, as discussed in Section 3.3, occlusion ratios. However, because of occlusion, segmentation may produce multiple objects for one complete object. Therefore, we merge objects that occlude each other and are close (occlusion ratio above 0.5 and centroid distance below 8cm) into one object and re-compute its occlusion ratio. Next, we compute object specific grasp (Eq. (2)) and observation probabilities (Eqs. (5) and (6)) using the occlusion ratios, observation history, and initially estimated parameters. We set the grasp prior count $n^{\text{prior}} = 0.5$. Finally, we make an observation if an object is dirty or clean based on the distance of the object color to precomputed color prototypes.

**Grasping.** Grasping an unknown object is performed by executing a top grasp, closing fingers around the centroid of the point cloud of the object to grasp, similar to e.g. [44].

**Estimating initial parameters**. Before actual experimental runs, we estimated experimentally the parameters of grasping and observation probability functions defined in Eqs. (3), (5), and (6). To estimate grasp parameters we attempted to lift cups positioned on the table using the robot arm, both when the cups were occluded and when not, and estimated grasp parameters ($\theta_{\text{G1}} = -0.904$, $\theta_{\text{G2}} = -0.087$) from the recorded success rates. For the occluded case we used the average occlusion ratio. We estimated observation function parameters for dirty ($\theta_{\text{D1}} = -0.895$, $\theta_{\text{D2}} = -0.087$) and clean ($\theta_{\text{C1}} = -0.193$, $\theta_{\text{C2}} = 0.0$) cups similarly, but instead of the lifting success rate, we used the observation success rate.

### *4.1. Experiments with simulated dynamics*

In multi-object manipulation, robot actions may have far reaching consequences: lifting first cup A and then cup B, may increase the probability of cup C being observed dirty from low to high by exposing it more fully. The robot has to consider at each time step, whether the information gain from lifting a cup yields more reward in the long run than executing an action which may yield higher immediate reward. Of course, because of the uncertainty in actions and observations, the real decision making problem can be even more complicated than this simple example implies. Consequently, our hypothesis is that a heuristic greedy manipulation approach is not sufficient and that planning several time steps into the future is needed. In order to study this hypothesis, we experimentally compared heuristic manipulation and the proposed POMDP approach with different planning horizons. Note that even though we simulate world dynamics, we estimate the grasp and observation probabilities using the physical robot arm and real observed occlusions. Moreover, we estimate the occlusions and locations of objects from point clouds captured by the Kinect sensor.

In the simulated dynamics experiments, we used ten different captured point clouds shown in Fig. 3 as the starting point for simulations. In these experiments, we form a world model from the point cloud and then repeatedly sample an initial belief and simulate the system using the probability model for 10 time steps. To get an initial belief, we sample particles using the cup dirtyness probability, which depends on past observations and which is defined in Eq. (4) (dirtyness is an object attribute). For evaluation purposes we also sample hidden object specific grasp success probabilities. In more detail, we sample for object $i$ the total amount of observed grasps $n_i = n_i^{\text{succ}} + n_i^{\text{fail}}$ from a Gamma probability distribution with shape 0.2 and scale 5.0, that is, a probability distribution where small $n_i$ are common, but also large $n_i$ are possible. We sample $n_i^{\text{succ}}$ from the uniform distribution between 0 and $n_i$, and keep $n_i^{\text{succ}}$ and $n_i^{\text{fail}}$ constant during each simulation run. Note that the magnitude of $n_i$ determines how much object specific grasp properties affect the grasp success probability compared to occlusion.

[1] For segmentation we use organized multiplane segmentation and organized Euclidean cluster extraction, part of the point cloud library http://www.pointclouds.org/.

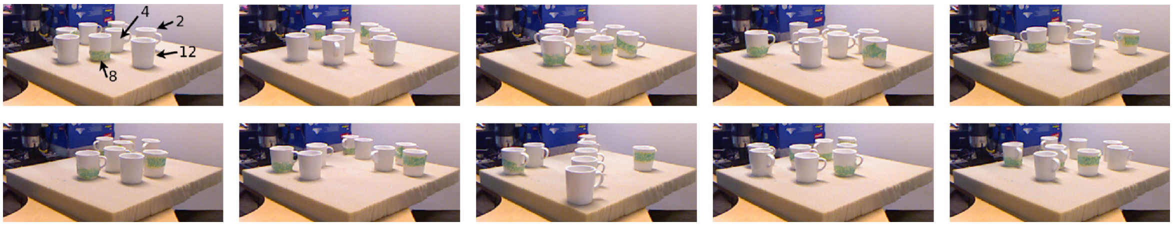


**Fig. 3.** Cropped kinect images of cup configurations used in the experiments. Each configuration contains four "dirty" (partly green color) and four "clean" cups. The green color on some cups is occluded. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

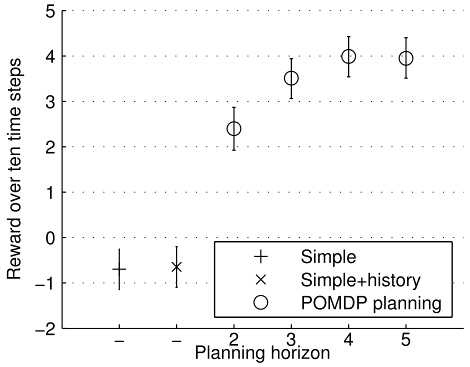


**Fig. 4.** The average reward sum and its 95% confidence interval (computed using bootstrapping) for the heuristic manipulation approach, heuristic manipulation approach utilizing grasp history information, and for the POMDP planning method.

### 4.1.1. Results

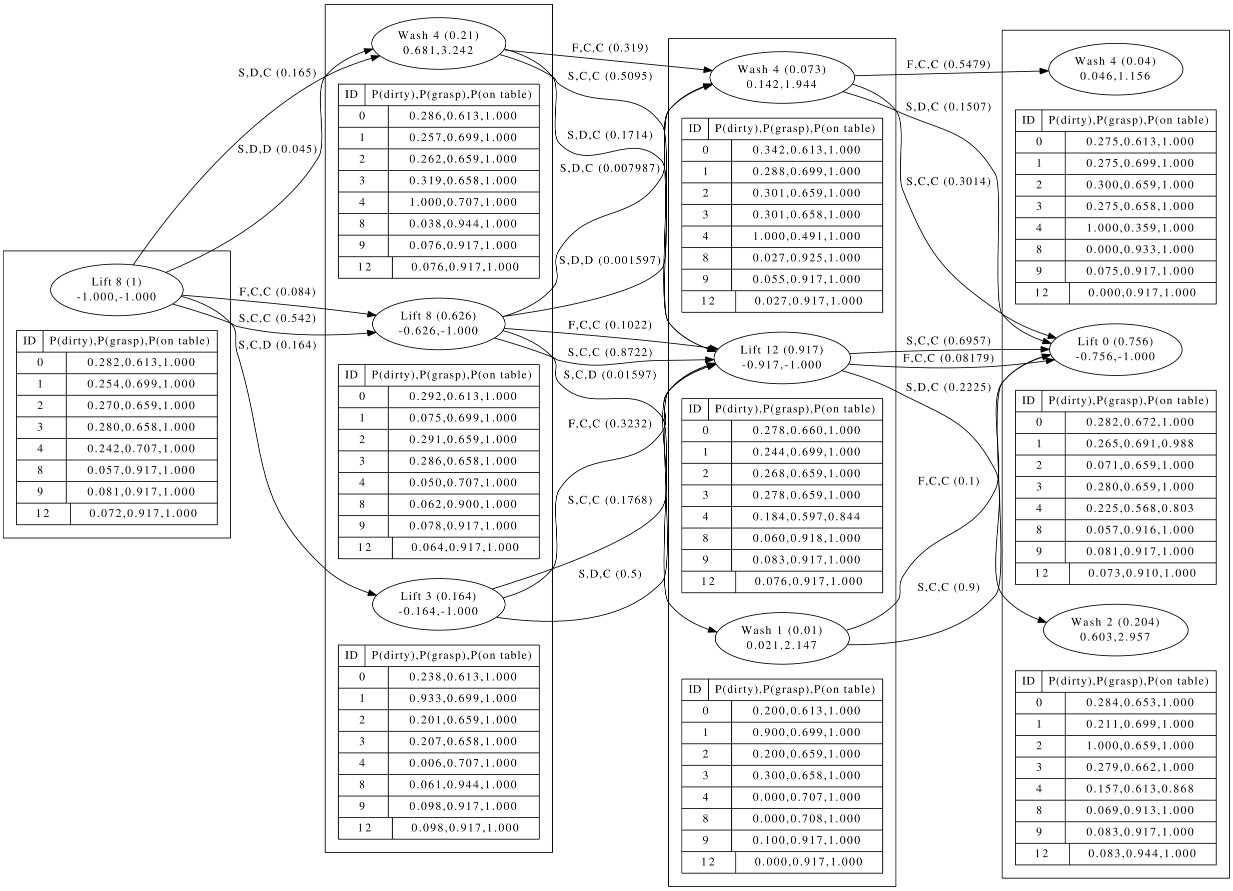
Fig. 4 compares POMDP planning with different planning horizons, ranging from two to five, with the heuristic manipulation approach. The POMDP policy graph had a width of three. Fig. 4 shows the average total reward over 100 simulation runs for each of the ten different cup configurations shown in Fig. 3. Overall, POMDP planning achieves higher reward than the heuristic manipulation approach. Interestingly, the performance difference between the heuristic approach with and without grasp history is not significant. To study this further, we ran over 2000 simulation runs for the scene shown in the third image, upper row, in Fig. 3. In this scene dirty cups are in front and thus the heuristic approach can select between several cups to move. Not surprisingly, the approach utilizing grasp history performed better (with non-overlapping average reward confidence intervals; not shown in Fig. 3).

It is also interesting that a POMDP planning horizon of three works significantly better than a horizon of two. Intuitively, one could imagine that short conditional plans, such as "lift a cup, and then, if the cup behind the lifted cup is dirty, move it into the dishwasher", would already perform very well. However, the results suggest that many problems require a complex policy to gain high reward. Fig. 5 shows a compact policy graph computed by the POMDP method for the first scene in Fig. 3. The policy illustrates information gathering through lifting cups, the effect of failed grasps, and complex conditional planning. In the policy graph, the agent lifts e.g. cups 8 and 12 (for reference, first RGB image in Fig. 3 shows cups 2, 4, 8, and 12) in order to gain information, and then when observing cups 4 or 2 as dirty, tries to move them into the dishwasher. In time step two, when the move of cup 4 into the dishwasher fails, the grasp probability of cup 4 decreases. In time step three, the agent tries to move cup 4 again. This highlights the important feature of principled uncertainty handling in POMDP planning. Even though grasping failed previously, the planner tries to move the same cup, because compared to the alternatives the grasp probability is still high enough.

We also tested different reward scenarios. Fig. 6 shows performance for the heuristic manipulation method and the POMDP method with a planning horizon of three for different reward choices. In the experiment, we varied the reward for lifting a cup/a failed grasp attempt and the reward for putting a clean object into the dishwasher. The POMDP method outperformed the heuristic method in each reward scenario. The reward for failed grasps/lifting a cup had a significant effect on the POMDP method's performance. One explanation is that when lifting cups becomes more expensive the benefit of planning over complex action–observation sequences decreases.

## 4.2. Robot arm experiments

In the previous section, we simulated world dynamics using a world model created from real robot grasps and point clouds captured by the visual sensor. In this section, we present experiments with a physical robot arm. In Section 4.2.1, we demonstrate crucial parts of our world model. In Section 4.2.2, we compare quantitatively the performance of the greedy



| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.282,0.613,1.000 |
| 1 | 0.254,0.699,1.000 |
| 2 | 0.270,0.659,1.000 |
| 3 | 0.280,0.658,1.000 |
| 4 | 0.242,0.707,1.000 |
| 8 | 0.057,0.917,1.000 |
| 9 | 0.081,0.917,1.000 |
| 12 | 0.072,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.286,0.613,1.000 |
| 1 | 0.257,0.699,1.000 |
| 2 | 0.262,0.659,1.000 |
| 3 | 0.319,0.658,1.000 |
| 4 | 1.000,0.707,1.000 |
| 8 | 0.038,0.944,1.000 |
| 9 | 0.076,0.917,1.000 |
| 12 | 0.076,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.292,0.613,1.000 |
| 1 | 0.075,0.699,1.000 |
| 2 | 0.291,0.659,1.000 |
| 3 | 0.286,0.658,1.000 |
| 4 | 0.050,0.707,1.000 |
| 8 | 0.062,0.900,1.000 |
| 9 | 0.078,0.917,1.000 |
| 12 | 0.064,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.238,0.613,1.000 |
| 1 | 0.933,0.699,1.000 |
| 2 | 0.201,0.659,1.000 |
| 3 | 0.207,0.658,1.000 |
| 4 | 0.006,0.707,1.000 |
| 8 | 0.061,0.944,1.000 |
| 9 | 0.098,0.917,1.000 |
| 12 | 0.098,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.342,0.613,1.000 |
| 1 | 0.288,0.699,1.000 |
| 2 | 0.301,0.659,1.000 |
| 3 | 0.301,0.658,1.000 |
| 4 | 1.000,0.491,1.000 |
| 8 | 0.027,0.925,1.000 |
| 9 | 0.055,0.917,1.000 |
| 12 | 0.027,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.278,0.660,1.000 |
| 1 | 0.244,0.699,1.000 |
| 2 | 0.268,0.659,1.000 |
| 3 | 0.278,0.659,1.000 |
| 4 | 0.184,0.597,0.844 |
| 8 | 0.060,0.918,1.000 |
| 9 | 0.083,0.917,1.000 |
| 12 | 0.076,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.200,0.613,1.000 |
| 1 | 0.900,0.699,1.000 |
| 2 | 0.200,0.659,1.000 |
| 3 | 0.300,0.658,1.000 |
| 4 | 0.000,0.707,1.000 |
| 8 | 0.000,0.708,1.000 |
| 9 | 0.100,0.917,1.000 |
| 12 | 0.000,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.275,0.613,1.000 |
| 1 | 0.275,0.699,1.000 |
| 2 | 0.300,0.659,1.000 |
| 3 | 0.275,0.658,1.000 |
| 4 | 1.000,0.359,1.000 |
| 8 | 0.000,0.933,1.000 |
| 9 | 0.075,0.917,1.000 |
| 12 | 0.000,0.917,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.282,0.672,1.000 |
| 1 | 0.265,0.691,0.988 |
| 2 | 0.071,0.659,1.000 |
| 3 | 0.280,0.659,1.000 |
| 4 | 0.225,0.568,0.803 |
| 8 | 0.057,0.916,1.000 |
| 9 | 0.081,0.917,1.000 |
| 12 | 0.073,0.910,1.000 |

| ID | P(dirty),P(grasp),P(on table) |
|---|---|
| 0 | 0.284,0.653,1.000 |
| 1 | 0.211,0.699,1.000 |
| 2 | 1.000,0.659,1.000 |
| 3 | 0.279,0.662,1.000 |
| 4 | 0.157,0.613,0.868 |
| 8 | 0.069,0.913,1.000 |
| 9 | 0.083,0.917,1.000 |
| 12 | 0.083,0.944,1.000 |

**Fig. 5.** A policy graph optimized by the POMDP method for four time steps, when starting execution from the configuration shown in the first point cloud in Fig. 3. At each time step an agent executes the action associated with the current graph node, makes an observation, and moves to the next layer node along the corresponding observation edge. Each graph node shows its action, the visiting probability in parenthesis, the expected reward, and the expected reward divided by the visiting probability. Each graph edge is labeled with the observation, that is, three symbols, e.g. "F,D,C", and a visiting probability in parenthesis. The first observation symbol denotes grasp success ("S") or failure ("F"); the second and third symbol denotes either dirty "D" or clean "C" for the first and second observed object, respectively. The box below a graph node displays for each object the dirtyness probability ("P(dirty)"), grasp success probability ("P(grasp)"), and the probability for the object to be on the table ("P(on table)"). Noteworthy: 1) failed grasps decrease the grasp probability, 2) lifting an object yields information about the dirtyness of objects behind the lifted object, 3) POMDP planning yields complex behavior.

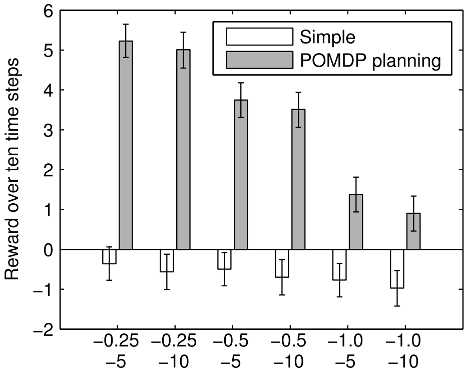
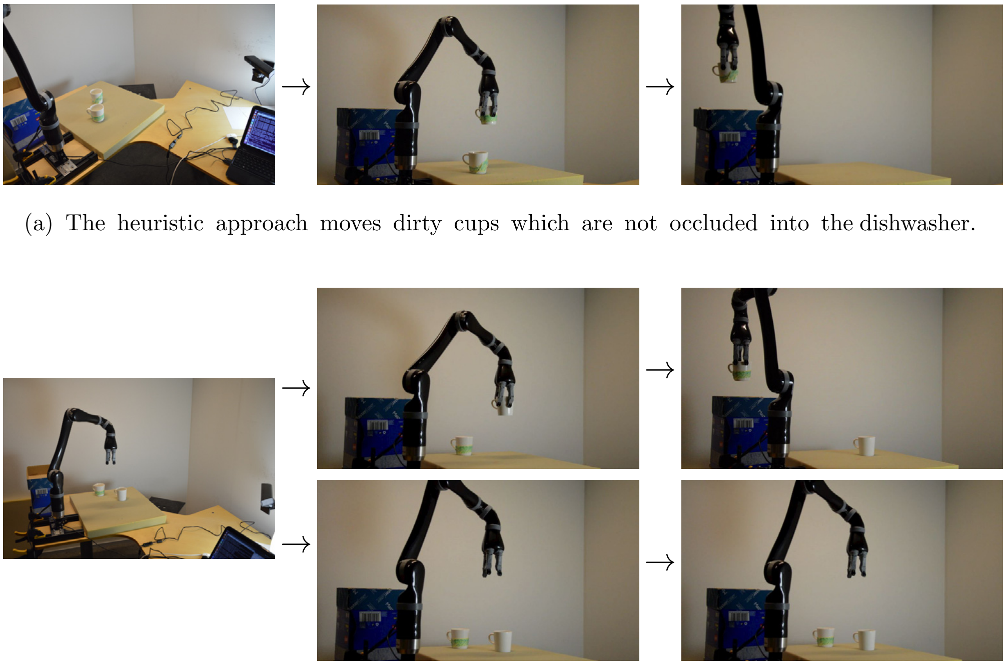


**Fig. 6.** The average reward sum and its 95% confidence interval (computed using bootstrapping) for the heuristic manipulation approach and for the POMDP planning method with a planning horizon of three for different reward scenarios. Each reward scenario has different rewards for moving a clean cup into the dishwasher (−5 or −10), and for lifting a cup/a failed grasp attempt (−0.25, −0.5, or −1.0).



(a) The heuristic approach moves dirty cups which are not occluded into the dishwasher.

(b) Because of occlusion the robot observes a dirty cup as clean. **Top:** In order to gain more information, the POMDP approach lifts the occluding cup, and then, when observing the dirty cup correctly, moves it to the dishwasher. **Bottom:** The heuristic approach executes the Finish action, because all cups appear clean.

**Fig. 7.** The robot tries to move possibly occluded dirty cups (partly green color) into the dishwasher (blue box). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

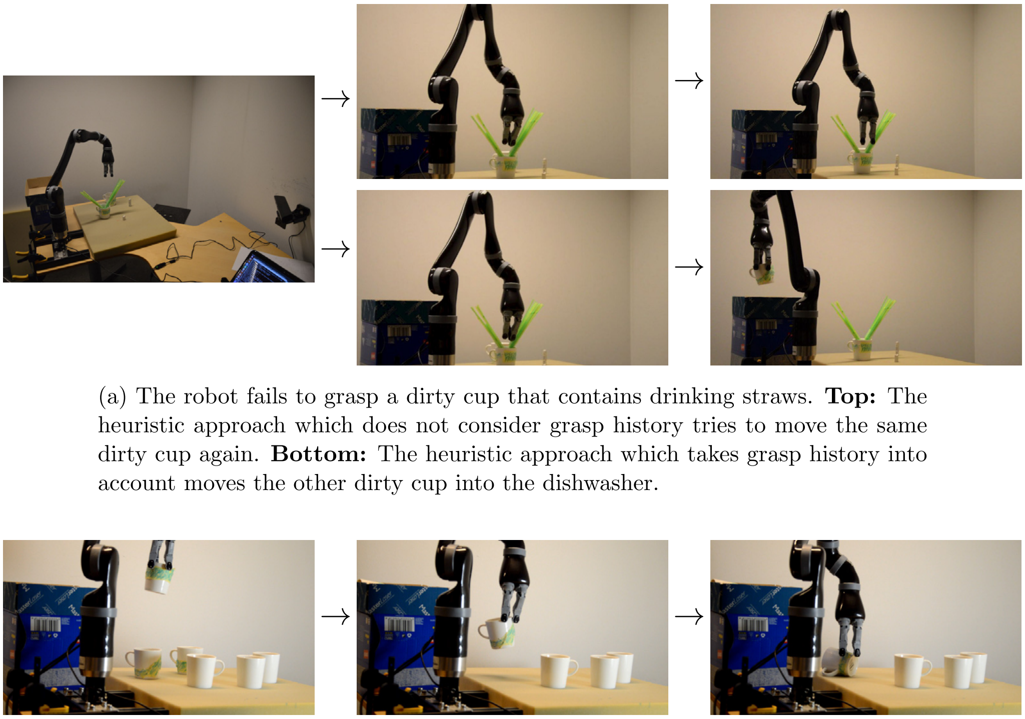
heuristic approach with the proposed POMDP approach. In the demonstrations, we show the usefulness of information gathering actions, such as lifting cups, in occluded settings. Furthermore, we experimentally investigate when object specific adaptive grasp probabilities are required. In addition, we examine in which situations the heuristic manipulation approach suffices for efficient operation, and when instead more comprehensive POMDP based decision making is required. The quantitative experiments show that the POMDP based approach significantly outperforms the simple greedy approach and yield insights, for instance, on why online planning is beneficial. Overall, the experiments show that real world problems require a model that takes occlusion into account, that multi-object manipulation problems require multi-step POMDP planning, and that adaptive action success probabilities are necessary in many situations.

We performed robot arm experiments using the Kinova Jaco arm. In the robot arm experiments, the Kinect sensor observes the scene, a method decides which action to execute, and then the robot arm executes the action. At each time step we estimate a belief from the captured point cloud and add the observation history information to this belief, to get the current belief. The method under evaluation decides on an action using the current belief. In order to maintain a consistent observation history and for detecting when a grasp succeeded or failed, we match current objects to objects in the previous time step: if an object is less than 4cm from its last spatial position, we assume it is the same object. If an object exists at the same location after it was moved or lifted, we assume the grasp failed.

### 4.2.1. *Demonstrations*

We claim that in multi-object manipulation, the robot may need to perform information gathering actions when objects are occluded, or when the grasp success probabilities of objects differ. However, when objects are in plain sight and easy to grasp decision making is easier. In this case, the problem requires no multi-step planning, and the heuristic policy of moving all cups that appear dirty into the dishwasher is sufficient. To test this, and to test whether our observation and state space models are applicable in physical robot arm experiments (we tested the model also in several other robot arm experiments which are discussed below), we performed robotic manipulation in a setup with dirty cups which are not occluded. Fig. 7a shows how the heuristic manipulation approach successfully moves the dirty cups into the dishwasher in this setup.

To test our occlusion model, and to test whether occlusion requires more complex decision making, we performed an experiment where the dirtyness of a cup is not apparent because another cup partly occludes the view on the dirty cup. The experiment in Fig. 7b demonstrates how the heuristic manipulation approach does not consider information gathering, and thus fails in the task. On the other hand, multi-step POMDP planning takes into account that the dirty cup may in fact be dirty, even though the robot observes it as clean, because the robot makes wrong observations on occluded cups with



(a) The robot fails to grasp a dirty cup that contains drinking straws. **Top:** The heuristic approach which does not consider grasp history tries to move the same dirty cup again. **Bottom:** The heuristic approach which takes grasp history into account moves the other dirty cup into the dishwasher.

(b) Grasping becomes harder. The robot has moved two dirty cups into the dishwasher, when another dirty cup drops onto the table and remains resting on its side. Following grasp attempts fail, because the cup is now more difficult to grasp.

**Fig. 8.** The robot tries to move dirty cups (partly green color) into the dishwasher (blue box). Some of the cups are harder to grasp than others. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a high probability. The POMDP approach lifts the clean cup, gains new information on the dirty cup, that is, observes the dirty cup as dirty, which increases the probability of the cup being actually dirty, and then successfully moves the dirty cup into the dishwasher.

Previously, we claimed that real world multi-object manipulation problems require an object specific adaptive grasp success probability. To test this claim and to verify that our adaptive grasp success model works, we performed an experiment with two dirty cups where the first cup is slightly occluded, and the second cup contains drinking straws that make correct grasping more difficult. The robot tries to move the second cup always first, because the occlusion on the first cup makes the initial grasp success of the second cup higher. For simplicity, we compared the heuristic manipulation approach with and without adaptive grasp success probabilities. As shown in Fig. 8a, both methods fail to grasp the second cup because of the drinking straws. The adaptive grasp success probability method updates the grasp success probability after observing a failed grasp, and moves the first dirty cup successfully into the dishwasher. The method that does not take grasp success history into account tries to grasp the same second cup again, even though an easier to grasp dirty cup would be available. These kind of situations occur often in practice. During experimentation with the robotic arm for example, as shown in Fig. 8b, the robot moves dirty cups, but when it moves the third dirty cup, the cup falls down and remains in a harder to grasp pose. We observed that when further grasps on the object failed, the grasp success probability decreased as expected.

### *4.2.2. Quantitative results*

In addition to the demonstrations, we performed a quantitative comparison between the simple greedy heuristic approach and the proposed POMDP approach in physical robot arm experiments. Similar to the experiments with simulated dynamics in Section 4.1, the goal was to move dirty, that is, partly green objects, into a "dishwasher". An object was observed dirty if the number of green pixels was at least 100.

Fig. 9 shows the four different scenes used. The fourth scene contains also toys to demonstrate the genericity of our approach. In each scene, we placed the objects on the table, and then ran the simple heuristic method and the POMDP method with a planning horizon of 3 after each other, five times each, yielding a total of twenty runs for each method over all four scenes. We reconstructed a scene after each run. Fig. 10 shows the results. Overall, the POMDP approach significantly outperformed the heuristic approach. Moreover, in each scene, the POMDP approach received higher rewards on average.

Regarding planning times, on a single low performance AMD A10-4600M CPU core the heuristic approach took on average 2 ms per time step while the POMDP approach took 2.6 s per time step. To put this in context, one timestep

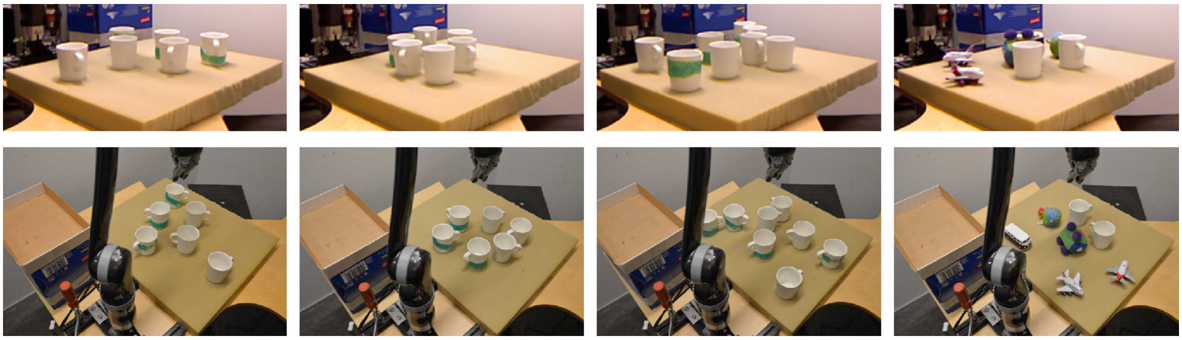
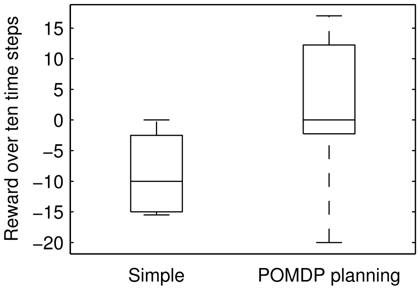


**Fig. 9. Top row**: cropped kinect images of the four scenes used in the robot arm experiments. **Bottom row**: corresponding photographs of the scenes. Each scene contains "dirty" (partly green color) and "clean" objects. Scenes one to three contain only cups but scene four contains also several toys. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



| Scene | Simple | POMDP |
|---|---|---|
| #1 | −9.1 | 11.6 |
| #2 | −15 | −4 |
| #3 | 0 | 2.6 |
| #4 | −10 | −1.7 |

**Fig. 10.** Robot arm experiments. We executed both the simple greedy approach "Simple" and the POMDP approach with a planning horizon of three "POMDP planning" five times in each of the four scenes shown in Fig. 9. **Left:** boxplot of the reward over ten time steps. "POMDP planning" was significantly better than "Simple" (the $p$-value was 0.00059 in the Mann–Whitney $U$ test [45]). **Right:** for both methods the average reward in each scene. "POMDP planning" had a larger average reward in each scene.

corresponding to moving an object from one location to another took on average 55 s so that the proportion of time spent on planning was 4.7% using the POMDP approach. Most of the time was taken by robot motion which was slow to guarantee safety. The visual capture and processing time was not optimized and was a few times the time of planning, typically around 10 s. Altogether, the time needed for the entire loading operation depended on the number of objects, being typically a few minutes.

Performance wise the heuristic approach was closest to the POMDP approach in scene 3. In scene 3, the two partly green objects closest to the Kinect were easy to grasp and the heuristic approach always successfully moved them to the dishwasher. Because of heavy occlusion the two partly green objects farthest from the Kinect were very hard to grasp. Therefore, while being usually able to move the easy to grasp objects, the POMDP approach had more difficulty in moving the other two partly green objects. Interestingly, among individual experiment runs, the POMDP approach had both the lowest (−20) and highest (17) reward. The lowest reward was possible because of the grasp and observation uncertainty, and because the POMDP approach was more active than the heuristic approach. Another interesting observation from the experiments was that occasionally an object could be dropped or tipped over. Our POMDP model does not explicitly take these kinds of events into account. However, in spite of this, the POMDP approach adapted to these unexpected situations because it always planned actions based on the belief estimated from current sensor readings.

### *4.3. Discussion*

The experiments confirm that multi-step POMDP planning is useful, when the order of actions is critical to the successful completion of the task. In particular, a POMDP estimates the value of information optimally. In an uncertain world, the probabilistic model used in POMDPs can weight different action choices in a principled manner. In contrast to a greedy approach, a POMDP may select actions that gather information, but do not yield immediate reward, when the problem so requires. In the multi-object manipulation experiments, the robot had to decide between lifting objects to gather information or moving objects that appear dirty into the dishwasher. Our POMDP model includes grasping success and learns grasping probabilities. Grasping unknown objects requires object specific grasp probabilities because each object may be different. However, even when predefined object models are available, adaptive object specific grasp probabilities may be useful; especially in heavily cluttered settings, with multiple objects, the large uncertainty about object pose and identity make grasping some objects harder than others and requires an adaptive approach.

The experimental setting in this paper was restricted to finding and moving dirty dishes into a dishwasher. However, a significant number of other applications, manipulation or otherwise, require information gathering or planning under uncertainty. When a robot is employed in an unstructured environment, for example, an ordinary home, the robot typically does not know the exact location of objects in advance and does not possess an exact model of the objects. These kinds of environments require reasoning about uncertainty and reasoning about when and how to obtain more information needed by the assigned task. Organizing objects, such as toys [46], is such a common task. The robot may need to find specific toys from several objects placed on top of a table or on the floor and put them into a box or on a shelf. Because of occlusions the robot does not see all objects completely and cannot be certain which object is which. Therefore, the robot has to consider information gathering actions. Moreover, in these kinds of settings grasp success is uncertain and often not known in advance.

As a concrete example, consider cleaning toys from a table into a box after a child has been playing. Successful identification and grasping of toys would depend on their visibility and the reward function could consist of a positive reward for putting a toy into the box and a negative reward for putting another object into the box. Another application is finding a particular object. For example, there could be several piles of objects and the robot has to plan how to remove objects from the piles, while taking into account how much information each removal will yield and how likely it is that the removal succeeds. A positive reward would be assigned to finding the object.

## 5. Conclusion

We presented a POMDP model for multi-object manipulation of unknown objects in a crowded environment. Because objects are occluded, their attributes are harder to observe and they are harder to manipulate. To address this, our POMDP model uses an *occlusion ratio* to define how much an object occludes another one. We use the occlusion ratio as a parameter in the observation and grasp probabilities of objects. In addition to occlusion specific grasp probabilities, our model also includes automatically adapting object specific grasp probabilities. To compute compact policies for the computationally complex POMDP model, we presented a new POMDP method that optimizes a policy graph using particle filtering. The method allows multi-step POMDP planning, both offline and online.

Experiments confirm that a heuristic greedy manipulation approach is not adequate for multi-object manipulation, but instead, the problem requires complex conditional multi-step POMDP plans that take long term effects into account. Moreover, object specific grasp probabilities are needed in many real-world situations.

In the future we plan to apply the presented POMDP model to other kinds of robotic tasks. Currently, we are extending the POMDP model to take into account the uncertainty in the composition of objects from segments. In general, to obtain true long-term autonomy, we believe that a robot should base its decisions on prior learned knowledge and adjust its world model to the specific environment it operates in. For this purpose a probabilistic Bayesian framework should be used that allows the robot to operate and learn in an uncertain, unstructured environment. In contrast to engineered solutions, learning offers the possibility to find solutions that generalize to unexpected situations and a possibility for autonomous adaptation. Our goal is an autonomous robot which can be placed in a complex new environment and which then knows how to adapt to the new environment. The work presented here is a step towards that goal.

## Acknowledgements

This work was supported by the Academy of Finland, decision 271394.

## References

[1] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1–2) (1998) 99–134.

[2] S. Thrun, Probabilistic robotics, Commun. ACM 45 (3) (2002) 52–57.

[3] J. Hoey, A. Von Bertoldi, P. Poupart, A. Mihailidis, Assisting persons with dementia during handwashing using a partially observable Markov decision process, in: Proceedings of the 5th International Conference on Vision Systems, ICVS, Bielefeld University Library, 2007.

[4] I. Chadès, E. McDonald-Madden, M.A. McCarthy, B. Wintle, M. Linkie, H.P. Possingham, When to stop managing or surveying cryptic threatened species, Proc. Natl. Acad. Sci. 105 (37) (2008) 13936–13940.

[5] J. Pajarinen, J. Peltonen, M.A. Uusitalo, A. Hottinen, Latent state models of primary user behavior for opportunistic spectrum access, in: Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, IEEE, 2009.

[6] C.H. Papadimitriou, J.N. Tsitsiklis, The complexity of Markov decision processes, Math. Oper. Res. 12 (1987) 441–450.

[7] T. Smith, R. Simmons, Point-based POMDP algorithms: improved analysis and implementation, in: Proceedings of the Twenty-First Annual Conference on Uncertainty in Artificial Intelligence, UAI, AUAI Press, 2005, pp. 542–549.

[8] H. Kurniawati, D. Hsu, W.S. Lee, SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces, in: Proceedings of Robotics: Science and Systems IV, MIT Press, 2008, pp. 65–72.

[9] D. Silver, J. Veness, Monte-Carlo planning in large POMDPs, in: J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, A. Culotta (Eds.), NIPS, in: Adv. Neural Inf. Process. Syst., vol. 23, Curran Associates Inc., 2011, pp. 2164–2172.

[10] H. Bai, D. Hsu, W. Lee, V. Ngo, Monte Carlo value iteration for continuous-state POMDPs, in: Algorithmic Foundations of Robotics IX, 2011, pp. 175–191.

[11] G. Shani, J. Pineau, R. Kaplow, A survey of point-based POMDP solvers, Auton. Agents Multi-Agent Syst. 27 (1) (2013) 1–51.

[12] P. Poupart, C. Boutilier, Value-directed compression of POMDPs, in: S. Becker, S. Thrun, K. Obermayer (Eds.), NIPS, in: Adv. Neural Inf. Process. Syst., vol. 15, MIT Press, 2003, pp. 1547–1554.

[13] D. McAllester, S. Singh, Approximate planning for factored POMDPs using belief state simplification, in: Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence, UAI, Morgan Kaufmann, 1999, pp. 409–417.

[14] J. Pajarinen, J. Peltonen, A. Hottinen, M. Uusitalo, Efficient planning in large POMDPs through policy graph based factorized approximations, in: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD, in: Lecture Notes in Computer Science, vol. 6323, Springer, 2010, pp. 1–16.

[15] P. Poupart, C. Boutilier, Bounded finite state controllers, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), NIPS, in: Adv. Neural Inf. Process. Syst., vol. 16, MIT Press, 2004, pp. 823–830.

[16] C. Amato, B. Bonet, S. Zilberstein, Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs, in: Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence, AAAI, AAAI Press, 2010.

[17] J. Pajarinen, J. Peltonen, Periodic finite state controllers for efficient POMDP and DEC-POMDP planning, in: Proceedings of the 25th Annual Conference on Neural Information Processing Systems, NIPS, 2011, pp. 2636–2644.

[18] S. Ross, J. Pineau, S. Paquet, B. Chaib-Draa, Online planning algorithms for POMDPs, J. Artif. Intell. Res. 32 (1) (2008) 663–704.

[19] S.S. Keerthi, E.G. Gilbert, Optimal infinite-horizon feedback laws for a class of constrained discrete-time systems: stability and moving-horizon approximations, J. Optim. Theory Appl. 57 (2) (1988) 265–293.

[20] J. Mattingley, Y. Wang, S. Boyd, Receding horizon control, IEEE Control Syst. 31 (3) (2011) 52–65.

[21] F. Doshi, J. Pineau, N. Roy, Reinforcement learning with limited reinforcement: using Bayes risk for active learning in POMDPs, in: Proceedings of the 25th International Conference on Machine Learning, ICML, ACM, 2008, pp. 256–263.

[22] P. Poupart, N. Vlassis, Model-based Bayesian reinforcement learning in partially observable domains, in: Proceedings of the Tenth International Symposium on Artificial Intelligence and Mathematics, ISAIM, 2008.

[23] S. Ross, J. Pineau, B. Chaib-Draa, P. Kreitmann, A Bayesian approach for learning and planning in partially observable Markov decision processes, J. Mach. Learn. Res. 12 (May) (2011) 1729–1770.

[24] S. Ross, B. Chaib-draa, J. Pineau, Bayesian reinforcement learning in continuous POMDPs with application to robot navigation, in: IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2008, pp. 2845–2851.

[25] H. Bai, D. Hsu, W.S. Lee, Planning how to learn, in: IEEE International Conference on Robotics and Automation, ICRA, 2013.

[26] T. Lozano-Pérez, M.T. Mason, R.H. Taylor, Automatic synthesis of fine-motion strategies for robots, Int. J. Robot. Res. 3 (1) (1984) 3–24.

[27] R.C. Brost, Automatic grasp planning in the presence of uncertainty, Int. J. Robot. Res. 7 (1) (1988) 3–17.

[28] S.M. LaValle, S. Hutchinson, An objective-based framework for motion planning under sensing and control uncertainties, Int. J. Robot. Res. 17 (1) (1998) 19–42.

[29] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006.

[30] L.P. Kaelbling, T. Lozano-Peres, Integrated task and motion planning in the now, Tech. Rep. MIT-CSAIL-TR-2012-018, MIT CSAIL, 2012.

[31] L.P. Kaelbling, T. Lozano-Peres, Integrated robot task and motion planning in belief space, Tech. Rep. MIT-CSAIL-TR-2012-019, MIT CSAIL, 2012.

[32] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.

[33] H. Kurniawati, Y. Du, D. Hsu, W.S. Lee, Motion planning under uncertainty for robotic tasks with long time horizons, Int. J. Robot. Res. 30 (3) (2011) 308–323.

[34] K. Hsiao, L.P. Kaelbling, T. Lozano-Peres, Grasping POMDPs, in: IEEE International Conference on Robotics and Automation, Rome, Italy, 2007.

[35] K. Hsiao, L.P. Kaelbling, T. Lozano-Pérez, Robust grasping under object pose uncertainty, Auton. Robots 31 (2–3) (2011) 253–268.

[36] K. Hsiao, M. Ciocarlie, P. Brook, Bayesian grasp planning, in: ICRA 2011 Workshop on Mobile Manipulation, 2011.

[37] J. Laaksonen, E. Nikandrova, V. Kyrki, Probabilistic sensor-based grasping, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Portugal, 2012, pp. 2019–2026.

[38] E. Nikandrova, J. Laaksonen, V. Kyrki, Towards informative sensor-based grasp planning, Robot. Auton. Syst. (2015), submitted for publication.

[39] M. Dogar, S. Srinivasa, A planning framework for non-prehensile manipulation under clutter and uncertainty, Auton. Robots 33 (3) (2012) 217–236.

[40] P. Monso, G. Alenya, C. Torras, POMDP approach to robotized clothes separation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 2012, pp. 1324–1329.

[41] D. Fischinger, M. Vincze, Y. Jiang, Learning grasps for unknown objects in cluttered scenes, in: IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013.

[42] S. Chakravorty, R. Erwin, Information space receding horizon control, in: IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL, IEEE, 2011, pp. 302–309.

[43] A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[44] J. Felip, J. Laaksonen, A. Morales, V. Kyrki, Manipulation primitives: a paradigm for abstraction and execution of grasping and manipulation tasks, Robot. Auton. Syst. 61 (3) (2013) 283–296.

[45] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1) (1947) 50–60.

[46] J. Pajarinen, V. Kyrki, Robotic manipulation in object composition space, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2014, pp. 1–6.