POMDP Manipulation via Trajectory Optimization

Ngo Anh Vien and Marc Toussaint

Abstract— **Efficient object manipulation based only on force feedback typically requires a plan of actively contact-seeking actions to reduce uncertainty over the true environmental model. In principle, that problem could be formulated as a full partially observable Markov decision process (POMDP) whose observations are sensed forces indicating the presence/absence of contacts with objects. Such a naive application leads to a very large POMDP with high-dimensional continuous state, action and observation spaces. Solving such large POMDPs is practically prohibitive. In other words, we are facing three challenging problems: 1) uncertainty over discontinuous contacts with objects; 2) high-dimensional continuous spaces; 3) optimization for not only trajectory cost but also execution time. As trajectory optimization is a powerful model-based method for motion generation, it can handle the last two issues effectively by computing locally optimal trajectories. This paper aims to integrate advantages of trajectory optimization into existing POMDP solvers. The full POMDP formulation is solved using sample-based approaches, where each sampled model is quickly evaluated via trajectory optimization instead of simulating a large number of rollouts. To further accelerate the solver, we propose to integrate temporal abstraction, i.e. macro actions or temporal actions, into the POMDP model. We demonstrate the proposed method on a simulated 7 DoF KUKA arm and a physical Willow Garage PR2 platform. The results show that our proposed method could effectively seek contacts in complex scenarios, and achieve near-optimal performance of path planing.**

I. INTRODUCTION

Robot manipulation in highly uncertain environments essentially requires information-gathering from interaction with objects and obstacles. Imagine a scenario in which a person manipulates objects without being equipped with vision perception ability. Entering a dark room, then after opening the door he wants to turn on the light. Based on previous experience, he has a certain belief that the light switch is on the wall next to the door. He would first want to seek contacts with the wall to disambiguate uncertainty about the position of the wall, then he would slide his hand along the wall to disambiguate the position of the light switch on the wall. All information-seeking actions use *tactile feedback*.

It is a very non-trivial problem as it has non-smooth dynamics due to contacts, and has high-dimensional and continuous state, control and observation spaces as innate properties of any robotic problems. Such problems can in principle be solved using decision-theoretic approaches such as POMDPs [1]. Once being formulated in this way, the uncertainty can be captured easily and disambiguated optimally through action execution. Efficient methods such as belief planning in Gaussian belief spaces using iLQG [2], [3] and information-gathering via tactile sensing [4] have achieved impressive results. However, those full POMDP solvers often require extensive computational time.

Recently, there has been remarkable work addressing similar problems via trajectory optimization, a family of powerful techniques for motion planning. Optimized trajectories are considered as reference for realtime execution controllers. With the presence of uncertainty, trajectories are often optimized to seek contacts with constraints [5], [6]. However their execution controllers are heuristically implemented in a non-trivial way.

This paper proposes an efficient approach for robot manipulation under uncertainty by integrating advantages from: trajectory optimization and POMDP. The contact-seeking motion problem is again formulated as a full POMDP. For leveraging POMDP solvers, we propose a new sample-based method that uses trajectory optimization to quickly evaluate each belief, instead of simulating a large number of rollouts. Specifically, our major contributions are:

- We propose the idea of factoring the POMDP model through contact observations. The **factored model** helps to separate parts of non-smooth dynamics from smooth ones. As a result, the evaluation of beliefs via Bellman equations are factored into value functions corresponding to parts of discrete dynamics and parts of continuous dynamics. The continuous dynamics part is perfectly suitable for trajectory optimization, and therefore it will be approximately marginalized in the Bellman equations of POMDP. The discrete part serves in one-step lookahead computation as normal Bellman equations.
- We exploit **temporal actions** [7], i.e. macro actions or options, that significantly accelerate the POMDP optimization process. We build a set of temporal actions merely by deriving policies from all sampled models. Each sample corresponds to one fully observable MDP model. Its policy can be computed quickly via trajectory optimization. The use of trajectory optimization is expected to reduce the computational time significantly.
- We use a **sample-based** method to solve the factored POMDP problem with a temporal action set. Each belief's value function is evaluated using Q-function approximation, or QMDP [8], instead of using a large number of simulations. QMDP value function of each sampled model will be estimated via trajectory optimization as discussed above. The POMDP policy is represented as a finite state controller and incrementally constructed via one-step lookahead.

Ngo Anh Vien and Marc Toussaint are with Machine Learning and Robotics Lab, University of Stuttgart, Germany, {vien.ngo;marc.toussaint}@ipvs.uni-stuttgart.de

This work was supported by the EU-ICT Project 3rdHand 610878.

The paper is organized as follows: the next section II reviews related work inspiring our approach. Section III revisits definitions of a) trajectory optimization for robot motion generation, and b) POMDP framework. We briefly discuss the advantages of the two techniques, and propose a factored POMDP model with a novel solver for it in Section IV. All experiments on a simulated KUKA arm and a physical PR2 robot platform are described in Section V. We conclude the paper with some remarks in Section VI.

II. Related Work

Manipulation through contacts or tactile feedback must be dated back to the work of Lozano-Perez [9] in which there was an exemplary peg-in-hole task. Recently this setting has received attention by teams participating in the DARPA ARM challenge [10]–[12] and the same study on humans when grasping without vision [13]. Those methods mostly depend on human expertise when designing the feedback controllers. Our proposed method approximately builds a finite state controller with less domain knowledge, using the principled POMDP framework.

Various approaches [6], [14]–[17] use optimization to rigorously build smooth contact models by approximating the expected reaction forces, e.g. using Linear Complementarity Problem (LCP) theory, constrained trajectory optimization involving contacts. Those methods have shown very impressive results in many complex sequential manipulation tasks of simulated robots. However they are not straightforward when implemented on physical robot platforms. In contrast, our method is easy to be implemented on simulated and physical robots.

The problem we are interested in is in principle a full POMDP which has previously been studied in many work such as in grasping [4], [18], or Gaussian belief planning using iLQG [2], [3], etc.. However such those approaches have very high computational complexity, and often require an exact model of the environment. Our proposed approach relaxes the requirement of an exact model, and circumvent the problem of computational complexity of full POMDPs, by combining the advantages of trajectory optimization and POMDP. As a result our resulting approach can inherit both ability in dealing with uncertainties from POMDP and the computationally efficient power from trajectory optimization.

III. Background

A. Trajectory Optimization

We use the Augmented Lagrangian method for trajectory optimization [19]. Let $x_t \in \mathbb{R}^n$ denote a joint configuration and $x_{t-k:t}$ a trajectory of length k from time $t-k$ to t. We formulate the trajectory optimization problem generally as a k-order optimization problem

$$\begin{aligned} \min_{x_{0:T}} \quad & \sum_{t=0}^{T} \phi_t(x_{t-k:t})^\top \phi_t(x_{t-k:t}) \\ \text{s.t.} \quad & \forall_t : \; g_t(x_t) \le 0 \; . \end{aligned} \tag{1}$$

where $\phi_t(x_{t-k:t}) \in \mathbb{R}^{n_t}$ are arbitrary k-order cost terms, and $g_t(x_t) \in \mathbb{R}^{m_t}$ are inequality constraints for each time t. There are plenty of ways to encode ϕ and g depending on different tasks and different optimizers. For example, square configuration space velocities are penalized simply using $\phi(x_{t-1}, x_t) \propto (x_t - x_{t-1})$, and the square accelerations using $\phi(x_{t-2}, x_{t-1}, x_t) \propto (x_t + x_{t-2} - 2x_{t-1})$. Similarly, for $k = 2$ we can penalize square torques or velocities or accelerations in any task space.

Toussaint et. al. [6] let the inequality constraints g_t represent contacts with objects. Then they used a modified Augmented Lagrangian (AL) [20] method to deal with inequality constraints. On the other hand, contact interactions are directly rewarded by appending a term $(-g(x_t) + \alpha)$ into each cost term $\phi(x_{t-k:t})$. This contact reward implies a squared potential $(-g(x_t) + \alpha)^2$ cost at each time slice t, which pulls towards a state violating the constraint with a positive value α. The resulting solution of this constrained optimization problem is a dual trajectory of (x_t, λ_t) which are joint configuration and Lagrange multiplier trajectories, where $\lambda_t \in \mathbb{R}^{m_t}$. This dual solution specifically encodes the temporal profile of contact interactions with constraints. In more detail, if one dimension's value of λ_t is positive, its respective constraint is active. During execution, a force controller then aims to reproduce this contact profile. This method obtains good results and is computationally efficient in both simulated and physical robot systems. However, the experiments are simple with a single uncertainty term as in the *hand-table* problem introduced in [6]. Though the formulation is general, the implementation of both optimizer and controller parts is non-trivial to extend to incorporate more than one uncertainty terms.

The purpose of all above modifications to constrained optimization is to simulate information-gathering behaviour via contacts similar to the way POMDPs do. Since the simple potentials do not capture uncertainty, the generated trajectory is not optimal in terms of a trade-off between exploration and exploitation. The constraints are released from contacts with the constraint plane at non-optimal time steps. We circumvent this problem by formulating this contact-seeking problem as a full POMDP. This POMDP problem will be solved computationally efficiently using sampling, where trajectory optimization plays a central role as a solver for each sampled instance.

B. Partially Observable Markov Decision Process

A POMDP $\mathcal{P}$ is defined as a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mathcal{C}, b_0\}$ [21]; where $\mathcal{S}$ is an unobservable state space; $\mathcal{O}$ is an observation space; $\mathcal{A}$ is a control space; $\mathcal{T}$ defines a state transition function, $\mathcal{T}(s, a, s') = \Pr(s'|s, a)$ that tells the probability of next states if take a control a at state s; $\mathcal{Z}$ is an observation function, $\mathcal{Z}(s', a, o) = \Pr(o|s', a)$ that defines the probability of observations at state s after taking control a; b_0 is an initial belief which represents a probability distribution over states $b_0(s) = \Pr(s_0 = s)$; and $\mathcal{C}$ is a cost function, $\mathcal{C}(s, a, s')$. A policy $\pi : \mathcal{B} \mapsto \mathcal{A}$ is defined as a mapping from the belief space to the control space.

The value function (or cost-to-go function) of a policy π is defined as the expected total return of discounted costs:

$$V^{\pi}(b) = E\Big[\sum_{t=0}^{\infty}\gamma^t c_t(s_t,a_t,s_{t+1})|b_0=b\Big] \tag{2}$$

if the agent starts from the belief b; where $\gamma \in [0,1]$ is a discount factor; and the expectation is w.r.t. stochasticity of $\pi, \mathcal{T}, \mathcal{Z}, \mathcal{C}, b_0$. The beliefs can be updated as $b_{t+1}(s) = \Pr(s_t = s|b_t,a_t,o_t)$. Abstractly, we denote $b_{t+1} = \mathcal{B}(b_t,a,o)$ as a belief update operator. The goal of the agent is to find an optimal policy π^* that minimizes the expected total return $V^{\pi^*}(b_0)$.

In object manipulation problems, the states are presumed to be $s \in \mathbb{R}^{n+n_e}$, where the robot's joint configuration is $x \in \mathbb{R}^n$, and the environment state is $e \in \mathbb{R}^{n_e}$ that is unobservable to the robot (i.e. the environment model, e.g. table position and size, object location). The control actions a are motor commands computed by the operational space/force controller. Observations $o = \{o_r, o_e\}$ contain a measurement vector sensed from the robot's internal state (noisy measurements at joints) and a measurement vector sensed from the external environment. Observations o_e are sensed forces of a F/T sensor at the wrist of the PR2 robot's arm. Alternatively, one can model observations o_e as binary feedback, i.e. contacts. Based only on a sequence of contacts, the robot should be able localize the table to accomplish his task. In other words, it is required to execute contact-seeking actions to gather information, in similar notion of POMDP planning, to accomplish the task. Alternatively, if we cast one type of binary observation (encoding contact or not) for each constraint, then we can implicitly represent Lagrange multipliers λ_t as in the constrained optimization approach by Toussaint et. al. [6]. A control sequence a_t (in execution controller) and λ_t can now be replaced by a sequence of control and observations in POMDP, $\tau = \{a_1, o_1, \cdots, a_T, o_T\}$. Therefore, in this paper we optimize a full POMDP controller instead of finding a dual trajectory. Our proposed POMDP solver is expected to be as much general as possible in term of continuous and very high-dimensional state and control spaces. In order to achieve these goals, our solver is based on sampling and exploits advantages from trajectory optimization. Each temporal action in our new POMDP formulation corresponds to one contact-seeking trajectory. As a trajectory can only be optimized via trajectory optimization if given the model, sampling from the belief is used to bridge between sample-based factored POMDP solvers and model-based trajectory optimization frameworks.

IV. Belief Planning via Trajectory Optimization

For simplicity, we assume that objects in the environment are static. We will discuss shortly how to generalize to the case of dynamic environments. We separate the state information $s \in \mathbb{R}^{n+n_e}$ into two types: the robot's joint state $x \in \mathbb{R}^n$, and the environment state $e \in \mathbb{R}^{n_e}$ (i.e. the environment model, e.g. table position and size, object location). We denote o_r the sensor information sensed from the robot's internal state, and o_e observations sensed from the environment's state. Observations from the robot's internal state could be understood as noisy information about its sensed force, sensed joint, and sensed position information. The noise comes from noisy sensors and imperfect actuators. Observations from the environment might come from the fact that the robot partially knows about its working environment. For instance, it is uncertain about the positions and poses of objects in the environment. In order to accomplish a given task, the robot must deal with uncertainty due to both its own imperfect mechanical subsystems, and external unknown environment.

A. Factored POMDP Model

We formulate the problem as a full POMDP as following. As dynamics of the environment is static, we can decompose the transition function $\mathcal{T}(s,a,s')$ as

$$\begin{aligned}\mathcal{T}(s,a,s') &= \mathcal{T}(x,e,a,x',e)\\ &= p(x',e|x,e,a)\\ &= p(x'|x,a,e)\end{aligned} \tag{3}$$

because e is static, where x and e are a state of the robot and environment, respectively. The factored observation function is

$$\begin{aligned}\mathcal{Z}(s,a,o) &= \mathcal{Z}(x,e,a,o_r,o_e)\\ &= p(o_r,o_e|x,e,a)\\ &= p(o_r|x,e,a)p(o_e|x,e,a)\end{aligned} \tag{4}$$

Though the full POMDP can be factored as above, solving it is known to be intractable. Because the state and control spaces are continuous and very high-dimensional, as for the case of a 22-DoF PR2 robot. To the best of our knowledge, there have been no efficient POMDP solvers able to deal with such large problems. Trajectory optimization is efficient and robust in generating motion trajectories if the environment model e is known. We propose a transformed POMDP that can enjoy many advantages of trajectory optimization with mild assumptions.

Given a model e (a sampled model from unknown environment is fully observable) we can use a trajectory optimization method to optimize a trajectory achieving the task, e.g. iLQG [22], differential dynamic programming [23], or constrained optimization [6], etc. Assuming that for a specific sampled model e and a starting joint configuration x_0, an optimized trajectory $\tau^e_{x_0}$ is returned. We denote by $\tau^e = \{\tau^e_{\cdot}\}$ the set of optimized trajectories with respect to all sampled starting joint configurations. We can represent τ^e approximately. This approximation is somehow similar to an MDP's value function approximation or QMDP [8], [24]. Because the set of optimized trajectories implicitly represents an optimal policy, an MDP solver or a stochastic optimal control solver is used. In our case, we use trajectory optimization to find an approximately optimal policy of the sampled model e. In detail, the QMDP approximation is

$$Q_{MDP}(b,a) = \int_{x,e} b(x,e)Q_{MDP}(x,e,a)dxde \tag{5}$$

where $Q_{MDP}(x, e, a)$ can be approximated by using a motion planning algorithm. Assuming this function is evaluated implicitly for the initial joint configuration, which is not shown due to brevity reason. By sampling from $b(x, e)$ we can approximate $Q_{MDP}(b, a)$ quickly. However, sampling from the joint belief $b(x, e)$ is redundant and might be intractable due to the high dimensionality of the joint space $\mathcal{X}$. Because the problem with noises from the robot's joint state can be efficiently dealt, e.g. by using a linear quadratic regulator (LQR) to execute the reference trajectory. Therefore if we know e accurately, we would simply use a traditional trajectory optimization algorithm to solve for a reference trajectory. Then LQG is used to regulate the execution of the optimized trajectory. Fortunately, in the case of manipulation problems, we may assume Gaussian noise on the control signals and we can use stochastic optimal control (SOC) methods such as iLQG or trajectory optimization (and a Laplace approximation around the optimal trajectory) to evaluate $Q_{MDP}(x, e, a)$ quickly and accurately. This can be done using the above POMDP factorization

$$\begin{aligned} Q_{MDP}(b, a) &= \int_e b(e) \int_x b(x|e) Q_{MDP}(x, e, a) dx de \\ &= \int_e b(e) Q_{SOC}(e, a) de \end{aligned} \tag{6}$$

where $b(x|e)$ is the belief over states given the true model e. The second integral expression can be evaluated without resorting to sampling. Using an SOC approach, the value function $Q_{SOC}(e, a)$ is computed when integrated out noisy states of the robot's joint configuration. To put it more simply, this is the total cost of a trajectory optimized from the initial joint configuration to the terminal state encoded by the hypothetical environment e.

Because the POMDP model is factored as above, we can use sample-based methods (sampling e from the prior) to solve for a near-optimal policy with respect to the use of SOC's value function as function approximation. However, a naive use of trajectory optimization in POMDP solvers can not efficiently cope with the following problems because: 1) the control and observation spaces are continuous; and 2) the real-time execution requires fast and smooth trajectories with a time resolution of a few milliseconds. In order to remedy those issues, we integrate temporal actions into the proposed approach. Temporal actions or macro-actions [7] have been widely used in POMDP solvers as a principled method to mitigate the curse of dimensionality and long-horizon problems. We posit that temporal actions are potentially helpful given their advantages in the case of very sparse observations, as in manipulation problems, e.g. grasping using hand-crafted macro actions [18], [25]. In the next subsection, we describe in detail how to integrate trajectory optimization and general temporal actions into a simple sample-based POMDP solver.

B. Transformed POMDP

We adopt the general POMDP formulation as in Section III-B to model contact uncertainties in manipulation problems. In order to integrate temporal actions and trajectory optimization, we first propose a transformed POMDP formulation whose action space consists of only macro actions. Each macro action is assumed to be one policy corresponding to one sampled model e as in Eq. 6. We call them QMDP macro actions or SOC macro actions (after integrating out noisy joint configuration states.

The transformed POMDP $\mathcal{P}'$ can be defined as tuple $\{\mathcal{S}, \mathcal{A}', \mathcal{O}, \mathcal{T}', \mathcal{Z}', \mathcal{C}', \gamma\}$, which has the same state and observation spaces; the action space $\mathcal{A}'$ consists of N actions $\{a^e_1, a^e_2, \cdots, a^e_N\}$, where a^e_i indicates that the robot is following a trajectory solved for the sampled model e_i. The termination condition for macro actions is when they observe the change in observations, e.g. when there is sensed forces or from sensed forces to NIL. Those macro actions are motivated from [18], [25] where they use just four direction macro actions. Each macro action is built as a solution of trajectory optimization for each sampled model. Therefore, we can use a large number of macro actions in order to generate smooth execution trajectories, similar to the way traditional sample-based POMDP solvers simulate a large number of samples to reduce estimation errors.

We use τ to denote a macro action representing, with a slight abuse of notation, either trajectories (each trajectory corresponds to one starting joint configuration) or a policy. If we write $\tau(x', x)$ it would define the next state x' if we start at state x and following the trajectory τ. If we write $\tau(x, u)$ it would define the policy of selecting a control u at state x. The transition function is $\mathcal{T}'(s, a^e_i, s') = \tau^{e_i}(x', x)$ that only affect the transition of joint states x. That function defines the probability of next joint configuration states if executing an action a^e_i that is an optimized trajectory w.r.t a sampled model e_i. The transition function might be deterministic or stochastic depending on the solution's representation of the trajectory optimization method. For instance, iLQG assumes Gaussian dynamics that would yield a stochastic transition $\tau^{e_i}(x', x, u)$, where u is the optimal control computed using feedback gains. Methods using constrained optimization like [6] would yield deterministic transitions. The observation function is $\mathcal{Z}'(s, a^e_i, o) = \sum_u \mathcal{Z}(s, \tau^{e_i}(s, u), o)$, if we assume that $\tau^{e_i}(s, u)$ is a "*policy*" solved for a sampled model e_i. Eq. 6 can be re-written and expanded using one-step lookahead for $\mathcal{P}'$ as

$$\begin{aligned} Q(x_0, a^e_i) &\approx \frac{1}{N} \sum_{i=1}^{N} \Bigg(c(x_0, a^e_i) + \\ &\sum_o \int_{x'} \mathcal{Z}'(x', a^e_i, o) \tau^{e_i}(x', x_0) Q_{MDP}(x', e^e_i, a^e_i) dx' \Bigg) \\ &= \frac{1}{N} \sum_{i=1}^{N} \Bigg(c(x_0, a^e_i) + \sum_o \mathcal{Z}'(x_1, a^e_i, o) Q_{SOC}(e^e_i, a^e_i) \Bigg) \end{aligned} \tag{7}$$

where x_0 is a starting joint configuration at a specific time slice; x_1 is assumed to be the next state in a deterministic trajectory; the function $c(x_0, a^e_i)$ that is the cumulative cost of executing an action a^e_i plays as an intermediate cost term.

Algorithm 1 FSC-Optimizer

```
1: Sample $N$ models $e_i \sim b_0$.
2: Given $x_0$ as an initial joint configuration.
3: Set a node $x_0$ as Root of $FSC$.
4: call FSC-EXPAND$(FSC, 0, x_0, E_0 = \{e_i\}_{i=1}^N)$
```

The second term in the last equation is the expectation of the approximate value function of the next belief set (the particles are filtered via the observation likelihood function $\mathcal{Z}$). We discretize observations as touch or no-touch depending on the sensed forces.

The formulation $\mathcal{P}'$ is approximation to $\mathcal{P}$ using temporal abstraction (temporal actions), sampling (sampled models) and function approximation (trajectory optimization or SOC). Therefore solving $\mathcal{P}'$ accurately would only solve $\mathcal{P}$ approximately. As each macro action is a QMDP (or SOC) policy as in Eq. 7, it could not gather information. However the policy derived recursively via one-step lookahead would do this task. In the next section, we propose a simple and efficient algorithm to solve $\mathcal{P}'$.

C. Trajectory Optimization Guided POMDP

First, we use a finite state controller (FSC) in order to represent the optimal policy for $\mathcal{P}'$, as its action and observation spaces are discrete. The FSC's nodes are annotated with actions and joint configurations at time t, (a_i^e, x_k), its edges are observations. The real execution from node (a_k, x_k) to node (a_{k+1}, x_{k+1}) is controlled by the trajectory solving the model a_k with starting state x_k and target x_{k+1}. Assuming that at time slice $t = 0$, the starting joint configuration is x_0 and the prior belief over environment is b_0, as presented in Algorithm 1. A root node is created and labeled with the initial joint configuration. Then it calls *FSC-EXPAND* to recursively construct a finite state controller, given a set of initial particles representing an initial belief. As shown in Algorithm 2, we use trajectory optimization as in line 3 to solve for policies for each sampled model given a starting state. Then one-step look-ahead as in Eq. 7 is applied to select the best action at the current belief node. This operation is expected to select actions able to do information gathering. Recursively, we build an FSC with the tree-depth of T in a breadth-first search style. Since observations are sparse due to involvement of few objects during manipulation, the FSC has a pretty small branching factor.

For environment has dynamic objects, we assume that their dynamics are parameterized by a parameter vector θ. The unknown parameter vector could be learnt through Reinforcement Learning (RL). One technique to use planning for learning is Bayesian RL [26]. In BRL, the unknown parameters are cast as unobservable state components in another POMDP [27], [28]. Then POMDP planning is used to find a Bayes-optimal policy that is known to be best trade-off between exploration and exploitation. An action hierarchy may also be used as in our recent work for offline learning [29], [30] or online learning [31].

Algorithm 2 FSC-EXPAND(FSC, k, node x, E_k)

```
1: if $(k = T)$ then
2:     return
3: end if
4: Set $x_k = x$    // a state at level $k$ of FSC
5: for each $e_i$ do
6:     $\tau^{e_i} = \mathbf{Traj\_Optim}(e_i, x_k)$.    // solve for a pol.
7:     Set $x_{k+1}^{e_i} = \tau^{e_i}[1]$    // next state in traj.
8: end for
9: Set $a_*^e, x_{k+1}^* = \arg\max_a Q_{MDP}(x, a_i^e)$ (Eq. 7)
10: Set action $a_*^e$ to node $(x_k)$
11: for each $o \in \mathcal{O}$ do
12:     Belief update: $E_{k+1} = \mathcal{B}(E_k, a_*^e, o)$
13:     Add a label $a_*^e$ to node $x_k$
14:     Create a node $x_{k+1}^*$ and connect to node $x_k$.
15:     Label the edge $x_k \to x_{k+1}^*$ with $o$.
16:     Call FSC-EXPAND$(FSC, k+1, x_{k+1}^*, E_{k+1})$
17: end for
```

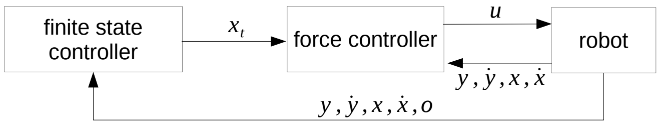

Fig. 1. Control Architecture

D. Execution Controller

We use an execution controller as shown in Figure 1 with two execution loops. The inner loop is an operational space force controller that computes motor commands. Its frequency is 1kHz. The outer loop is an optimized finite state controller that gives targets at each time slice and has a lower frequency. We use a similar force controller as described in [6].

V. Experiments

We evaluate the proposed approach in two tasks on both simulated and physical robots as illustrated in Fig. 2. The first task, *hand-table* problem, as studied in [6] requires a robot to find a trajectory to reach a point somewhere 10cm above the table. Through finding contacts with the table, the robot would gain certainty first about the height of table, then release the table's plane for the hanged target. This task has one uncertainty, which is the height of the table. The second task mimics the *peg-in-hole-like* problem. In this task, the robot must both find the table's height and size before reaching the center of the table. The robot is expected to slide its end-effector on the table to find one of the table's edge, then slide back to the table's center.

A. Simulated KUKA Arm

1) Hand-Table Problem: In the first task, we compare our POMDP approach against a constrained optimization method, i.e *dual-execution* in [6], and against an adaptive submodularity approach [32]. The setting of the dual-execution method is similar to its original paper. For both *dual-execution* and POMDP approaches, we optimize their

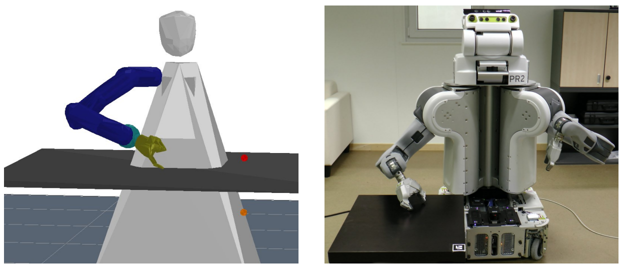

Fig. 2. A simulated KUKA arm and a physical Willow-Garage PR2.

trajectory and policy over 200 time slices. On a 32-bit, 8GB Ram, 2.9Ghz$\times 4$ computer, the computation of our method is 40.94 seconds for 100 samples, and the constrained optimization method from [6] is 0.743 seconds. The adaptive submodularity application for touch-based manipulation is recently introduced [32] and has shown remarkable results. We generate 100 linear motions that moves downward on the z-axis from different starting states until sensing a contact (with the table). Each linear motion is one trajectory optimization problem using the optimizer from Toussaint et. al. [6].

The result of exemplary trajectories for each algorithm is reported in Fig. 3. There are three major points in this result.

- The POMDP controller can deal with uncertainties, therefore its trajectory only slides on table little enough to gain certainty. In our setting, we observe that the uncertainty is totally disambiguated after at least 15 consecutive touches. Once gaining certainty, the robot just moves directly to the goal. As the *dual-execution* method sets *funneling* reward manually, the constrained optimization trajectory would also prefer contacts if it is highly weighted. Therefore, the *dual-execution*'s trajectory would not optimize when it should release the table.
- The behaviour of the POMDP and adaptive submodularity approaches looks comparable. As the adaptive submodularity approach can also seek contacts with the table, it optimizes a trajectory going to the estimated goal after getting certainty from the contacts. However the adaptive submodularity approach is not for path planning, therefore its entire trajectory from start to goal is suboptimal in terms of total cost (e.g. shortest path). Our approach based on POMDP planning can apparently do path planning under uncertainty in a principled way, and hence outperform both heuristic methods like dual-execution and non-path planning methods like adaptive submodularity.
- The POMDP controller can estimate the height of the table via their observation branching of the finite state controller, therefore forces are softly exerted after it has touched the table for the first time. Meanwhile, the *dual-execution* policy still pushes forces so harshly that the trajectory has sometime penetrated into the table.

2) Peg-In-Hole-like Problem: In the second task, the robot is uncertain over both the table's height (in z-axis) and

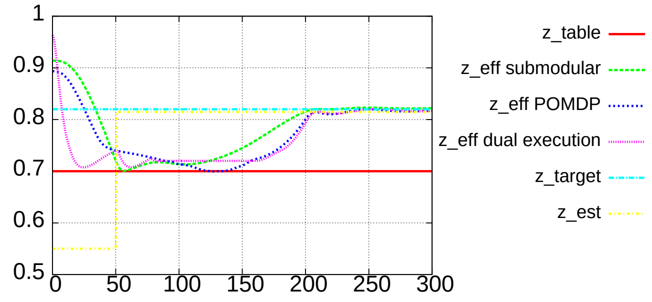

Fig. 3. Exemplary trajectories of the dual-execution, POMDP, and adaptive submodularity algorithms (x-axis is time steps). y-axis shows the z coordinates (in meters). The legends with $z\_$ mean the z coordinates of: _eff: end-effector, _table: true table's height, _est: the current estimate.

table's position (in x-axis). The *dual-execution* method can not solve this task straightforwardly, as it requires to design the contact plane in an ad-hoc way. Even if we work hard and succeed in designing such an objective function, it is still difficult to design the execution controller that at the same time maintains contacts with the table plane, and pulls the end-effector into the edge plane constraint. Therefore, we skip comparisons against the *dual-execution* method, and only report results from our method.

In Fig. 4, we report a typical run from a finite state controller optimized with 100 samples in approximately 118 seconds. The behaviour can be described as: first the robot gradually goes down to find the first contact with the table; after that it is still uncertain about the center position, therefore it slides to either edges in oder to localize the table's position (assuming that it knows the table's size).

B. On Willow Garage PR2 Robot Platform

We re-implemented all above experiments on the Willow Garage PR2 robot platform. Contacts are indicated by sensed forces returned by the force/torque sensor at the wrist.

1) Hand-Table Problem: We samples 100 times from the prior over the environment's model, i.e. height information, the computational time on this full DoF robot is approximately 300 seconds. Figure 5 shows two sequences of a run from two algorithms: the *dual-exection* approach and our proposed method. As seen in the figure, the 4th picture shows the difference of behaviour between two algorithms. At that time slice, our controller tells the robot to release the contact, meanwhile the *dual-exection* controller tells the robot to keep sliding.

2) Peg-In-Hole-like Problem: We samples 200 times from the prior over the environment's model, i.e. height and length information, the computational time on this full DoF robot is approximately 800 seconds. On the second task, the results are illustrated as in Fig. 4. We can also see the similar behaviour of the simulated KUKA arm's experiment in PR2's experiment. See the attached video for better explanation!

VI. Conclusion

In this paper, we have proposed a simple and efficient belief planning method for robotic manipulation problems.

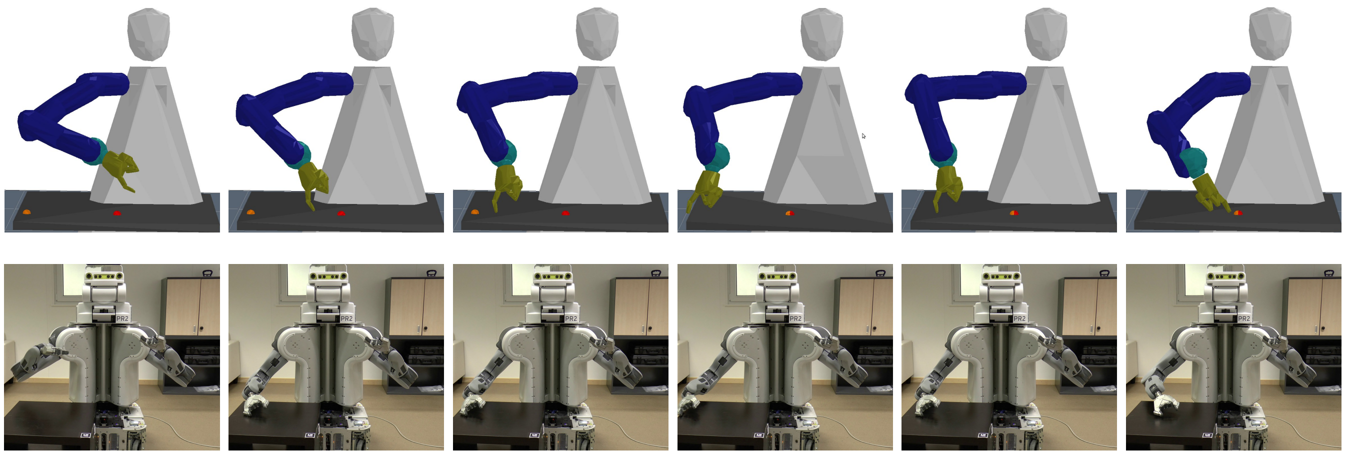

Fig. 4. A typical run of the peg-in-hole-like task on the simulated KUKA robot arm (top-row pictures) and the Willow Garage PR2 platform (bottom-row pictures) (numbering: left to right): 1) initial pose, 2) it's going down to detect the table's height, 3) sliding right to its estimate target (an orange point), 4) detected the right edge, 5) updated the target, slide back to the newly updated target, 6) reached the true target (a red point).

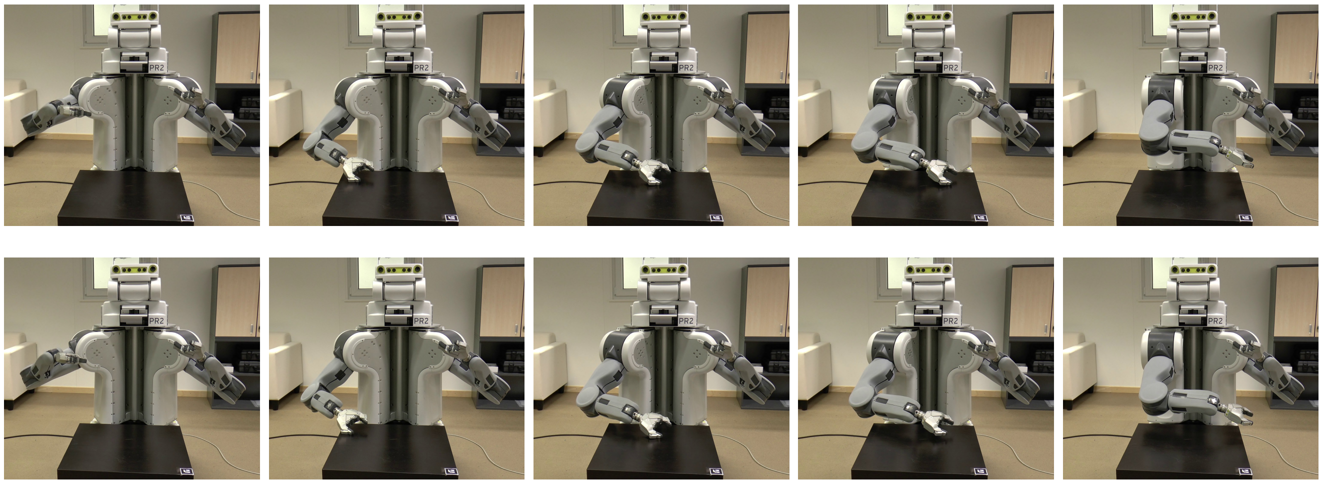

Fig. 5. Exemplary runs of the first task on the Willow Garage PR2 platform. The top row are pictures from the *dual-execution* method. The bottom row are pictures from our approach. Each row is numbered from left to right: 1) initial pose; 2) first contact with the table; 3) start sliding on the table; 4) top row: still sliding, bottom row: start to release contact; 5) top row: still sliding, bottom row: keep going to the target without sliding; 6) reached the target.

We exploit tactile feedback to formulate the problems as a full POMDP. To overcome the intractability in solving such high-dimensional POMDPs, we have integrated powerful trajectory optimization into a sample-based POMDP solver, to replace the traditional expensive simulation techniques. To further accelerate the solver, we adopt the idea of using temporal actions whose termination condition is based on sensed forces. As a synergy effect of such integrated solution, we have obtained a powerful framework with solving time increases approximately linearly in the number of samples, due to sparse observations in manipulation problems.

REFERENCES

[1] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *International Journal of Robotic Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

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

[3] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[4] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Robust grasping under object pose uncertainty," *Autonomous Robots*, vol. 31, no. 2-3, pp. 253–268, 2011.

[5] I. Mordatch, E. Todorov, and Z. Popovic, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, p. 43, 2012.

[6] M. Toussaint, N. Ratliff, J. Bohg, L. Righetti, P. Englert, and S. Schaal, "Dual execution of optimized contact interaction trajectories," in *IROS*, 2014.

[7] J. Pineau, "Tractable Planning Under Uncertainty: Exploiting Structure," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, 2004.

[8] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *J. Artif. Intell. Res. (JAIR)*, vol. 13, pp. 33–94, 2000.

[9] T. Lozano-Pérez, M. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, 1984.

[10] N. Hudson, T. Howard, J. Ma, A. Jain, M. Bajracharya, S. Myint, C. Kuo, L. Matthies, P. Backes, P. Hebert, T. J. Fuchs, and J. W. Burdick, "End-to-end dexterous manipulation with deliberate interac-

tive estimation," in *IEEE International Conference on Robotics and Automation, ICRA*, 2012, pp. 2371–2378.

[11] M. Kazemi, J. Valois, J. A. Bagnell, and N. S. Pollard, "Robust object grasping using force compliant motion primitives," in *Robotics: Science and Systems*, 2012.

[12] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. Voorhies, G. S. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Auton. Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[13] R. Deimel, C. Eppner, J. lvarez Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," in *International Symposium on Robotics Research (ISRR)*, 2013.

[14] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic Foundations of Robotics X - Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2012*, 2012, pp. 527–542.

[15] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *IEEE International Conference on Robotics and Automation, ICRA*, 2011, pp. 1071–1076.

[16] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2012, pp. 4914–4919.

[17] I. Mordatch, Z. Popovic, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the 2012 Eurographics/ACM SIGGRAPH Symposium on Computer Animation, SCA*, 2012, pp. 137–144.

[18] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," in *WAFR*, 2010, pp. 175–191.

[19] M. Toussaint, "Newton methods for k-order Markov constrained motion problems," *CoRR*, vol. abs/1407.0414, 2014.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[21] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, 1998.

[22] W. Li and E. Todorov, "An iterative optimal control and estimation design for nonlinear stochastic system," in *Decision and Control, 2006 45th IEEE Conference on*. IEEE, 2006, pp. 3242–3247.

[23] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. Elsevier, 1970.

[24] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, 1995, pp. 362–370.

[25] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Grasping pomdps," in *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, 2007, pp. 4685–4692.

[26] H. Bai, D. Hsu, and W. S. Lee, "Planning how to learn," in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2013, pp. 2853–2859.

[27] P. Poupart, N. A. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete bayesian reinforcement learning," in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, 2006, pp. 697–704.

[28] N. A. Vien and W. Ertel, "Monte Carlo tree search for Bayesian reinforcement learning," in *11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 1*, 2012, pp. 138–143.

[29] N. A. Vien, H. Ngo, and E. Wolfgang, "Monte carlo bayesian hierarchical reinforcement learning," in *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, 2014, pp. 1551–1552.

[30] N. A. Vien, H. Q. Ngo, S. Lee, and T. Chung, "Approximate planning for bayesian hierarchical reinforcement learning," *Appl. Intell.*, vol. 41, no. 3, pp. 808–819, 2014.

[31] N. A. Vien and M. Toussaint, "Hierarchical Monte-Carlo planning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2015, pp. 3613–3619.

[32] S. Javdani, M. Klingensmith, J. A. Bagnell, N. S. Pollard, and S. S. Srinivasa, "Efficient touch based localization through submodularity," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013, pp. 1828–1835.