Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

# Controlling Contact-Rich Manipulation Under Partial Observability

Florian Wirnshofer*, Philipp S. Schmitt*, Georg v. Wichert*, and Wolfram Burgard†
*Siemens Corporate Technology, Siemens AG †Department of Computer Science, University of Freiburg, Germany

***Abstract*—In this paper, we present an integrated, model-based system for state estimation and control in dynamic manipulation tasks with partial observability. We track a belief over the system state using a particle filter from which we extract a *Gaussian Mixture Model* (GMM). This compressed representation of the belief is used to automatically create a discrete set of goal-directed motion controllers. A reinforcement learning agent then switches between these motion controllers in real-time to accomplish the manipulation task. The proposed system closes the loop from joint sensor feedback to high-frequency, acceleration-limited position commands, thus eliminating the need for pre- and post-processing. We evaluate our approach with respect to five distinct manipulation tasks from the domains of active localization, grasping under uncertainty, assembly, and non-prehensile object manipulation. Extensive simulations demonstrate that the hierarchical policy actively exploits the uncertainty information encoded in the compressed belief. Finally, we validate the proposed method on a real-world robot.**

## I. Introduction

In industrial applications, robotic systems are used for various tasks that involve object manipulation. Examples include material handling, logistics, and assembly. However, industrial working environments are highly structured, e. g., using fixtures or part feeders. In less structured environments manipulation still poses a considerable challenge. The robot must keep track of its current knowledge about the environment and obtain new information from sensors. Perceiving the object with external sensors such as cameras is challenging since a direct view on all relevant aspects of the scene may be hindered, especially during direct interaction with the object. In such cases, the robot must make use of internal sensory feedback available in contact.

Consider the example depicted in Figure 1, where a "blind" robot must push an object into a goal region. The robot is only given a coarse prior on the object's initial position. In such a setting, the robot must consider the risk that comes with performing actions. A rash action based on a coarse prior could cause the object to be pushed out of reach or even off the table. On the other hand, it is important that the robot performs its task quickly and purposefully. The robot should be equipped with the ability to naturally balance efforts between localization (exploration) and more goal-directed behaviors (exploitation) and thus base its course of action on the present degree of uncertainty. This type of manipulation problem can be modeled as a *Partially Observable Markov Decision Process* (POMDP) with the following properties:

The presented research is financed by the TransFit project which is funded by the German Federal Ministry of Economics and Technology (BMWi), grant no. 50RA1701, 50RA1702, and 50RA1703.

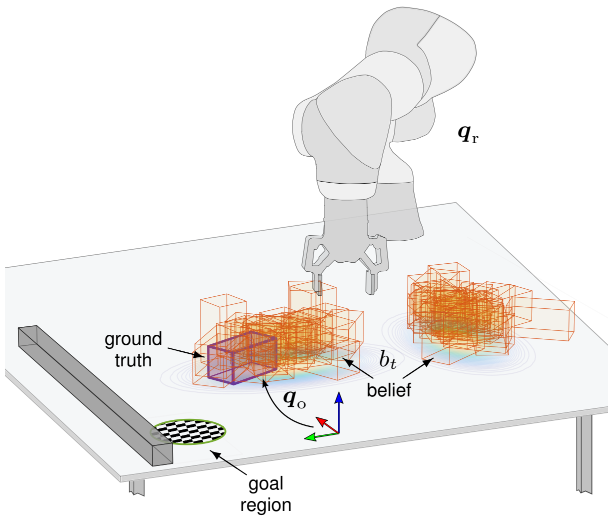


Fig. 1. A robot must push an object into a goal region. No visual feedback is available and the robot is only given a coarse multi-modal prior $b_{t=0}$ about the object location $q_o$. To achieve the task at hand the robot must track and improve the estimate using measurements available in contact. The robot must balance between information gathering and goal-directed actions. To prevent the object from being pushed off the table, the robot must consider the risk of actions under the present degree of uncertainty.

1) **High-dimensional state and action space**: Robot, environment, and the object interact through forces in various ways, e. g., through grasping or non-prehensile pushing. The robot may knock over the object, push it out of its reachable workspace, or even off the table. To account for this rich scope of possible interactions and the various (potentially unrecoverable) outcomes, one must consider the high-dimensional, continuous state space including positions and velocities of both robot and object.
2) **Nonlinear dynamics and multi-modal distributions**: Contact dynamics are highly nonlinear, exhibit abrupt discontinuities, and tend to quickly cause the formation of multi-modal distributions across the system state.
3) **Real-time and real-world constraints**: Robots have restricted workspaces, configuration-dependent dynamical limits, and require high-frequency actuation signals. Object manipulation involves dynamic transitions that require real-time tracking and instantaneous reactions.

High-dimensional continuous state and action spaces, complex dynamics, and real-time requirements render this POMDP, i. e., the problem of optimal action selection for contact-rich manipulation under partial observability, a major challenge.

The contribution of this work is an integrated system that solves challenging real-world manipulation problems given coarse priors on the initial state and with only joint-encoder readings as feedback. We propose a hierarchical policy, where an upper, long-term decision layer switches between high-frequency, goal-directed motion controllers based on a compressed representation of the current belief. Our system considers the full state space of objects and takes into account the robot's dynamic and kinematic limits. We evaluate the control framework in simulation on five different manipulation tasks and on a real-world robot.

## II. Related Work

In the following, we review existing approaches to solving a POMDP from the domains of sampling-based motion planning (Section II-A) and control-related approximate iterative techniques (Section II-B). Some of these algorithms have been used and extended in approaches directly targeting contact-rich manipulation (Section II-C).

### A. Motion Planning Under Uncertainty

To account for actuation and sensing uncertainties in dynamical systems with continuous state, action, and observation spaces, the works of Prentice and Roy [31], Van Den Berg et al. [38], Agha-Mohammadi et al. [2], and Bry and Roy [5] extend sampling-based roadmap- and tree-planners to *beliefs* (distributions over states). Platt et al. [28] and Van Den Berg et al. [39] use iterative trajectory optimization to obtain locally optimal control policies. To efficiently handle the POMDP planning problem, these methods assume an unimodal Gaussian belief. Unfortunately, the dynamics in contact quickly form multi-modal beliefs. Hauser [11] and Platt et al. [29] remove this assumption by using a particle-based belief representation and online-replanning. Using replanning schemes in our application context is challenging, since non-prehensile manipulation involves computationally demanding dynamic models yet requires real-time, reactive behaviors.

### B. Approximate POMDP Solvers

A substantial share of work addresses the POMDP problem using approximate, point-based value iteration. Established solvers including the works of Pineau et al. [27], Porta et al. [30], Kurniawati et al. [22], or Bai et al. [4] approximate policies for a set of probable scenarios offline and use these policies during execution. However, these methods require an adequate discretization of state, action, and observation space and fall victim to the curse of dimensionality on typical problems of contact-rich manipulation. The online solvers of Kurniawati and Yadav [21], Seiler et al. [34], and Chen et al. [6] address continuous POMDPs. Recent discrete online-solvers [10] have shown to handle large state and observation spaces. However, these methods perform a substantial number of policy-rollouts during runtime, which, given the computation-intense contact dynamics, prohibits fast, reactive behaviors. To obtain an offline policy for a continuous POMDP, Thrun [35] uses sampled off-policy rollouts of a simulated particle filter. We combine this methodology with belief augmentation [32] and use a deep neural network to efficiently approximate the Q-function [42]. Other works including Egorov [8], Hausknecht and Stone [12], and Le et al. [23] have made use of deep neural networks to tackle partial observability. However, these works are tailored to visual feedback and noise in the form of image flickering or partial image occlusions.

### C. Action Generation for Manipulation Under Uncertainty

Kaelbling and Lozano-Pérez [16] and Phiquepal and Toussaint [26] address combined task and motion planning under partial observability using a high-level tree search. They formalize actions on an abstract level (e. g., *pick*, *look*, *place*) and branch based on symbolic observations. Instead, our method directly processes the immediate joint-encoder feedback and, without additional post-processing, provides high-frequency joint-servo commands.

Hsiao et al. [14] introduces a POMDP framework for grasping static objects. Notably, the works of Koval et al. [20, 19] provide near-optimal POMDP policies for non-prehensile pushing tasks. However, the above methods require an a priori state space discretization and assume a quasistatic interaction model. Under quasistaticity, objects stop moving as soon as the robot halts or breaks contact with the object. In contrast, the method proposed in this work simulates the full dynamics of both robot and object. As a consequence, the object can tip over, roll off, or even drop off a table. Our method can thus implicitly acquire measures to avoid risky maneuvers that lead to such potentially unrecoverable outcomes.

## III. Problem Formulation & Background

In the following, we give a brief overview on the POMDP (Section III-A) and place it in context of contact-rich manipulation (Section III-B).

### A. POMDP

We consider a POMDP with a continuous state $x \in \mathcal{X}$, actions $u$, and continuous, noisy observations $z$. On executing an action, the robot experiences a reward $r$. In solving POMDPs, we aim to find a policy $\Pi$ that maximizes the expected cumulative reward. We refer to specific points in time using the subscripts $t$ and $t' = t + 1$.

The state transition density $p\left(x_{t'} \mid x_t,\, u_{t \to t'}\right)$ accounts for the stochastic nature of motion in unstructured environments. Specifically, $p\left(x_{t'} \mid x_t,\, u_{t \to t'}\right)$ describes the probability of ending up in $x_{t'}$, when starting in $x_t$ and applying action $u_{t \to t'}$ between $t$ and $t'$. The probability of subsequently measuring $z_{t'}$ is given by the measurement density $p\left(z_{t'} \mid x_{t'}\right)$.

*Bayesian Belief Update:* Given its recorded history of actions and observations, the robot can track its current belief about the system state

$$b_t = p(x_t \mid \underbrace{u_{0 \to 1},\, \dots,\, u_{t-1 \to t},\, z_1,\, z_2,\, \dots,\, z_t}_{\text{history}}).$$

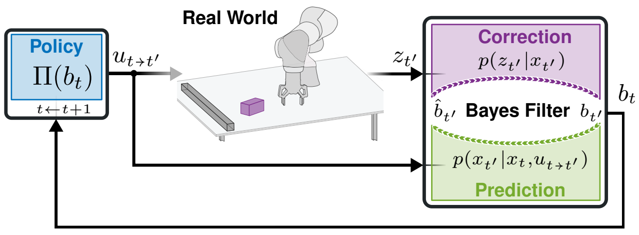


Fig. 2. Manipulation under uncertainty through Bayes filter belief updates (estimation) and *Belief* MDP decision-making.

To avoid the treatment of a possibly unbounded history (especially in the context of learning a mapping from belief to action), it is common practice to draw on the recursive belief update of a Bayes filter

$$\begin{aligned}\hat{b}_{t'} &= \int_{\mathcal{X}} b_t \, p\left(x_{t'} \mid x_t, \, u_{t\to t'}\right) \mathrm{d}x_t \\ b_{t'} &= \eta \, p\left(z_{t'} \mid x_{t'}\right) \hat{b}_{t'} .\end{aligned} \tag{1}$$

Here, $\eta$ is a normalizer that ensures $\int_{\mathcal{X}} b \; \mathrm{d}x = 1$. In doing so, we can break our initial problem of decision-making under uncertainty into two parts: recursive state estimation (i. e., tracking the belief) and executing a learned *Belief MDP* policy, which maps the current belief $b_t$ to the next action $u_{t\to t'}$. Refer to Figure 2 for a schematic overview of this decision-making procedure.

*Belief MDP:* Starting from an initial belief $b_0$, the fundamental goal now lies in finding a belief MDP policy $\Pi\left(b_t\right)$ that maximizes the cumulative expected reward

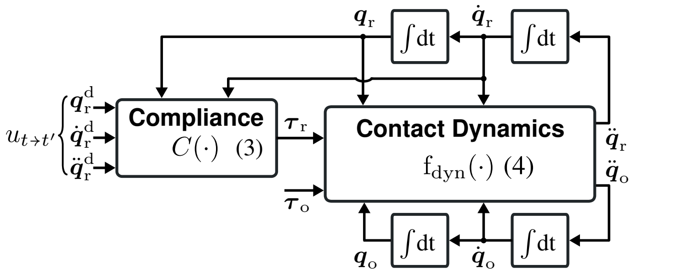
$$R^{\pi}\left(b_t\right) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} \, r_{t+\tau} \mid u_{t+\tau\to t'+\tau} = \Pi\left(b_{t+\tau}\right)\right],$$

with discount factor $\gamma \in [0, \, 1)$.

## B. POMDP for Contact-Rich Manipulation

In the following, we put the previously established POMDP framework into context with contact-rich manipulation. The approach to modeling contact-rich manipulation under uncertainty is based on our previous work [43]. The typical setting considered within the present work consists of a robot and a single object with known geometry and dynamics. An example of such a system is depicted in Figure 1. Here, a robot has to move an object to a desired location, despite being given only a coarse prior on where the object initially rests. For the remainder of this work, we abbreviate all robot- and object-related variables with subscript $(\cdot)_{\mathrm{r}}$ and $(\cdot)_{\mathrm{o}}$, respectively. Furthermore, we consider system variables to be stacked vectors of robot and object variables, e. g., the system configuration $\boldsymbol{q} = \left[\boldsymbol{q}_{\mathrm{r}}, \boldsymbol{q}_{\mathrm{o}}\right]^{\top}$ contains the robot's configuration $\boldsymbol{q}_{\mathrm{r}} \in \mathbb{R}^{n_{\mathrm{r}}}$ followed by the object configuration $\boldsymbol{q}_{\mathrm{o}} \in \mathrm{SE}\,(3)$.

*State, Action, & Measurement:* In contact-rich manipulation tasks, the system state at time $t$ is given as $x_t = \left(\boldsymbol{q}, \, \dot{\boldsymbol{q}}\right)_t$, i. e., the system configuration $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$. As feedback $z_t$,



$u_{t\to t'}$
$\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}$
$\dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$
$\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$
Compliance
$C(\cdot)$ (3)
$\boldsymbol{\tau}_{\mathrm{r}}$
$\boldsymbol{\tau}_{\mathrm{o}}$
Contact Dynamics
$\mathrm{f}_{\mathrm{dyn}}(\cdot)$ (4)
$\ddot{\boldsymbol{q}}_{\mathrm{r}}$
$\ddot{\boldsymbol{q}}_{\mathrm{o}}$
$\boldsymbol{q}_{\mathrm{r}}$
$\dot{\boldsymbol{q}}_{\mathrm{r}}$
$\boldsymbol{q}_{\mathrm{o}}$
$\dot{\boldsymbol{q}}_{\mathrm{o}}$
$\int \mathrm{dt}$

Fig. 3. Combined model of contact dynamics and compliance controller. The system input is given as a sequence of acceleration-bounded set-points $(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}, \, \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}})$. The compliance controller allows to specify inputs on the level of positions while ensuring physically plausible and safe interaction under uncertainty.

we only consider noisy measurements of the robot's joint position $\boldsymbol{q}_{\mathrm{r}}$. The corresponding measurement model is

$$p\left(z_{t'} \mid x_{t'}\right) = \mathcal{N}\left(\boldsymbol{q}_{\mathrm{r}\,t'}, \, \Sigma_{\mathrm{enc}}\right), \tag{2}$$

where $\Sigma_{\mathrm{enc}}$ denotes the assumed measurement variance of the axis position encoders. Note that this choice of measurement model does not rely on any object related quantities.

Robots are typically commanded via joint trajectories. Hence, we assume the inputs $u_{t\to t'}$ to be an acceleration-bounded sequence of set-points $(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}, \, \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}})$. To ensure a safe interaction between the robot and its (potentially unobserved) environment we generate the immediate robot torques $\boldsymbol{\tau}_{\mathrm{r}} \in \mathbb{R}^{n_{\mathrm{r}}}$ by means of a stable compliance controller as shown in Figure 3. Once the robot enters contact with the environment, the compliance controller exerts torques

$$\boldsymbol{\tau}_{\mathrm{r}} = C\left(\boldsymbol{q}_{\mathrm{r}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}, \, \boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}, \, \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}\right), \tag{3}$$

based on the deviation of the current robot position/velocity from the desired reference point $(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}, \, \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}})$. The controller typically imposes a spring-damper alike behavior. When no interaction occurs, the robot precisely tracks the reference.

*Motion Model:* Given the current state $x_t$ and an input $u_{t\to t'}$, we can compute the state $x_{t'}$ through numerical integration of the underlying contact-based forward dynamics

$$\begin{aligned}\ddot{\boldsymbol{q}} &= \mathrm{f}_{\mathrm{dyn}}\left(\boldsymbol{q}, \, \dot{\boldsymbol{q}}, \boldsymbol{\tau}_u\right) \\ &= \boldsymbol{M}\left(\boldsymbol{q}\right)^{-1}\left(\boldsymbol{\tau}_u + \boldsymbol{\tau}_c - \boldsymbol{c}\left(\boldsymbol{q}, \, \dot{\boldsymbol{q}}\right) - \boldsymbol{g}\left(\boldsymbol{q}\right)\right).\end{aligned} \tag{4}$$

Here, $\ddot{\boldsymbol{q}} \in \mathbb{R}^{n_q}$ with $n_q = n_{\mathrm{r}} + 6$ denotes the acceleration of the system and $\boldsymbol{M} \in \mathbb{R}^{n_q \times n_q}$ is the mass matrix. Vectors $\boldsymbol{c} \in \mathbb{R}^{n_q}$ and $\boldsymbol{g} \in \mathbb{R}^{n_q}$ are the Coriolis and gravitational forces, respectively. Vector $\boldsymbol{\tau}_c \in \mathbb{R}^{n_q}$ summarizes all effects arising from contact. We denote the vector of input torques as $\boldsymbol{\tau}_u = \left[\boldsymbol{\tau}_{\mathrm{r}}, \boldsymbol{\tau}_{\mathrm{o}}\right]^{\top}$, where again, the robot torques $\boldsymbol{\tau}_{\mathrm{r}}$ are generated by means of a compliance controller (3) and a commanded reference point $(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}, \, \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}, \, \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}) \mathrel{\widehat{=}} u_{t\to t'}$.

This model of dynamics itself is deterministic. To account for both random behavior and model errors, we require a probabilistic model of motion.

As motion model we propagate the state according to the compliance-interfaced contact dynamics model of Figure 3. To

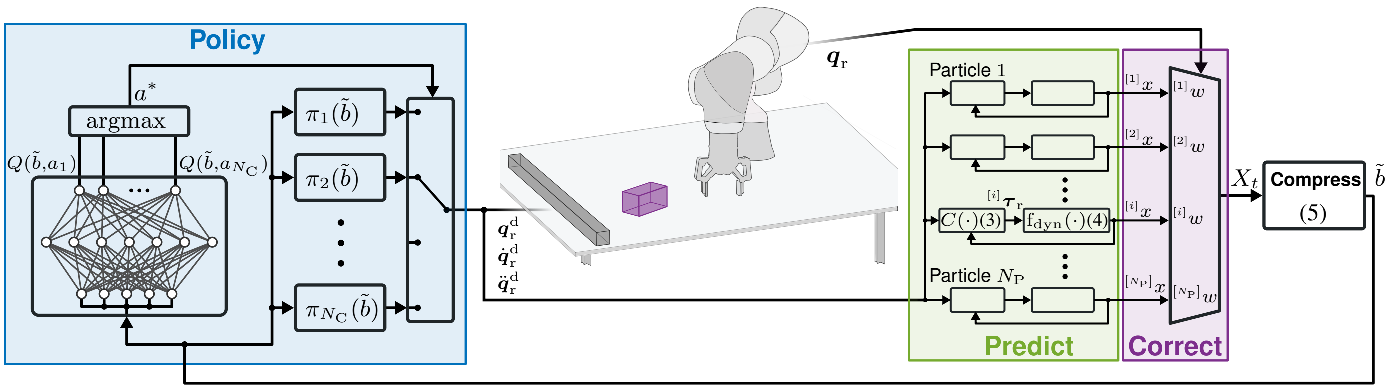


Fig. 4. Integrated system for manipulation under uncertainty. A particle filter for contact-rich manipulation tracks the current belief. The set of particles is compressed to a GMM, which then serves as an input to a hierarchical policy. The upper decision layer acts as situation classifier. Based on the current belief, i. e., the present degree of uncertainty, the upper layer selects a high-frequency local motion controller. The lower layer, that is, a set of goal-directed local motion controllers, provides the high-frequency, acceleration- and velocity-bounded reference to the robot's compliance controller.

systematically add uncertainty to the model without violating physical consistency, we add force noise to the object through the object torques $\boldsymbol{\tau}_o \in \mathbb{R}^6$. We generate these random forces acting on the object according to an Ornstein-Uhlenbeck stochastic process [37]. For further details on the motion model, we refer to [43].

## IV. Controlling Manipulation Under Uncertainty

In order to systematically solve the POMDP of the previous section we need two main components: an approximation to a Bayes filter to track the current belief $b_t$ and a policy $u_{t \to t'} = \Pi(b_t)$. In this work we focus entirely on the policy $\Pi$.

Due to the nonlinearity of contact dynamics, we choose to approximate the Bayes filter using a particle filter. We use the particle filter introduced in [43] to obtain a high-frequency (200 Hz) estimate of the current belief. The resulting problem is now to map the particle belief onto an executable robot command in real-time that efficiently solves the underlying manipulation task. We address this problem with a hierarchical approach.

In a first step, we compress the high-dimensional input to our policy, i. e., the particle set, into a lower-dimensional representation of the belief by fitting a *Gaussian Mixture Model* (GMM) into the distribution. In a second step, we derive a set of Cartesian motion controllers from this mixture model that are goal-directed with respect to the manipulation task. Given this compressed belief and set of motion controllers, the manipulation task is reformulated as a switching control problem. In this problem, a high-level controller switches at a low control frequency between high-frequency motion controllers.

The low-frequency upper layer can be viewed as situation classifier, taking into account the present uncertainty. The high-frequency controllers enable locally reactive behaviors and provide the reference command for the compliance controlled robot. The switching control problem is then solved via reinforcement learning.

### A. Belief Compression

The filter's belief estimate is represented as a set of $N_P$ weighted $({}^{[i]}w)$ particles. Unfortunately, this particle set $b_t \approx X_t = \left\{ \langle {}^{[i]}x_t, {}^{[i]}w_t \rangle \mid i = 1, \ldots, N_P \right\}$, forms a high-dimensional input with $N_P \dim(x_t)$ ($\gtrsim 10\,000$) entries. To obtain a lower-dimensional statistic of this input that is also independent of the number of particles, we compress the particle belief using a GMM over configurations:

$$b_t \approx \sum_{i=1}^{N_G} h_i \mathcal{N}(\boldsymbol{q}_t;\, \mu_i,\, \Sigma_i)\,.$$

Here, each of the $N_G$ modes is parameterized by means of a heft $h_i$, a mean $\mu_i = [\mu_{r,i}, \mu_{o,i}]^\top$, and a covariance $\Sigma_i$. The overall compressed belief is therefore given as

$$\tilde{b} = \langle h_1, \mu_1, \Sigma_1, \ldots, h_{N_G}, \mu_{N_G}, \Sigma_{N_G} \rangle\,. \tag{5}$$

This lower-dimensional representation is useful for two reasons: it is a suitable input to a neural network that forms the high-level controller of our approach and it is the basis for a task-oriented synthesis of the low-level controllers introduced in the following section.

### B. Low-Level, Goal-Directed Motion Controllers

Imagine the task of pushing an object on a table from the left side to the right. On an abstract level a manipulator might move above the object to avoid collisions, then move to the left-hand side of the object and push it to the right. The low-level controllers of this section are intended to provide such goal-directed behaviors to our system.

As shown in Figure 5, we define a set of Cartesian relative positions between the end-effector of the manipulator and the object. For each object mode $\mu_{o,i} \in \mathrm{SE}(3)$ of our GMM representation of the belief, we determine $n$ Cartesian end-effector targets $\mathbf{t}_j \in \mathrm{SE}(3)$. This results in $N_C = n\, N_G$ Cartesian goals. To actually steer the end-effector towards such a Cartesian goal, we automatically construct $N_C$ controllers $\pi_k(\tilde{b})$ using an acceleration-resolved variant [33] of the eTaSL/eTC control framework [1]. This works as follows:

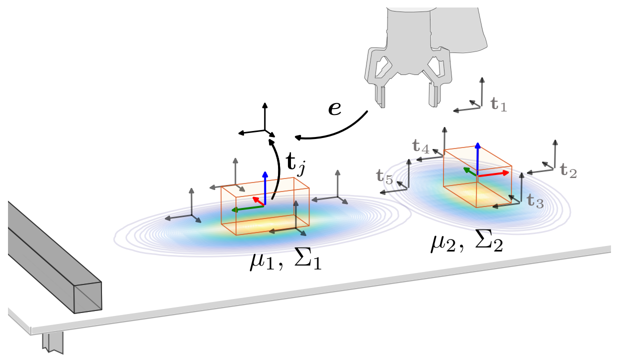


Fig. 5. The goal-directed motion controllers act based on the current GMM. Each motion controller represents an attractor for the end-effector. The attractor is encoded as transform $\mathbf{t}_j \in \mathrm{SE}\,(3)$ relative to one of the mixture modes. The controller imposes zero-convergent critically damped dynamics on the deviation $\boldsymbol{e}$ between end-effector and local target.

The deviation between the current end-effector position and target $\mu_{\mathrm{o},i}\,\mathbf{t}_j \in \mathrm{SE}\,(3)$ is measured via a vector-valued error function $\boldsymbol{e}(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}},\, \mu_{\mathrm{o},i}\,\mathbf{t}_j)$. For this error function $\boldsymbol{e}$ we define stable and properly damped second order dynamics

$$\ddot{\mathbf{e}} = \mathbf{J}_e \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} + \dot{\mathbf{J}}_e \dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} \overset{!}{=} -\mathbf{K}_{\mathrm{p}}\,\boldsymbol{e} - \mathbf{K}_{\mathrm{d}}\dot{\boldsymbol{e}}, \tag{6}$$

where $\mathbf{J}_e$ is the Jacobian of the error function $\boldsymbol{e}\,(\cdot)$. In this, and the following context, $\mathbf{K}_{\mathrm{p}}$ and $\mathbf{K}_{\mathrm{d}}$ denote diagonal gain and damping matrices of fitting dimension. To impose limits on both joint positions and velocities, we introduce the vector-valued inequality constraints

$$\boldsymbol{c}_p(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}) \leq \underline{\mathbf{0}} \qquad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{c}_v(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}) \leq \underline{\mathbf{0}}.$$

Similar to the error function $\boldsymbol{e}$, we impose stable, critically damped dynamics for both constraint functions.

The desired acceleration $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$ is now determined via the following optimization problem:

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}} \quad & \|\ddot{\mathbf{e}} - (-\mathbf{K}_{\mathrm{p}}\,\boldsymbol{e} - \mathbf{K}_{\mathrm{d}}\dot{\boldsymbol{e}})\|_2 \\ \text{s.t.} \quad & \frac{\partial \ddot{\boldsymbol{c}}_p}{\partial \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}}\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} \leq -\mathbf{K}_{\mathrm{p}}\,\boldsymbol{c}_p - \mathbf{K}_{\mathrm{d}}\dot{\boldsymbol{c}}_p \\ & \frac{\partial \ddot{\boldsymbol{c}}_v}{\partial \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}}\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} \leq -\mathbf{K}_{\mathrm{d}}\dot{\boldsymbol{c}}_v \\ & \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\min} \leq \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} \leq \ddot{\boldsymbol{q}}_{\mathrm{r}}^{\max}, \end{aligned} \tag{7}$$

where $\boldsymbol{c}_*$, $\dot{\boldsymbol{c}}_*$, and $\ddot{\boldsymbol{c}}_*$ are the value and time derivatives for given $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}}$ and $\dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$. Vectors $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\min}$ and $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\max}$ denote the lower and upper bounds on the reference acceleration $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$. All constraints are linear in $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$ and the cost function is linear quadratic in $\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}}$. The resulting quadratic program can therefore be solved efficiently with established numerical solvers.

The controllers introduced above are acceleration-resolved. We can thus switch among the controllers without causing discontinuities in the robot's motion. Switching acceleration-resolved controllers has another notable advantage. The controllers act as attractors that accelerate the end-effector towards a desired Cartesian target, but only for short periods of time. By continually changing the direction of acceleration, we can create a rich set of movements from a sparse grid of coarsely specified motion controllers. The placement of the targets $\mathbf{t}_j$, though coarse, requires some hand-engineering. A simple yet suitable heuristic is to place the targets such that the the object can be approached from every side, yet motions between neighboring targets do not interfere with the object.

### C. High-Level Manipulation Controller

Given a compressed representation of the belief and a corresponding set of low-level motion controllers, we may now reformulate the manipulation problem as a switching control problem. Based on the current compressed belief $\tilde{b}$, a high-level controller selects one of the motion controllers at discrete time intervals. The output of this controller is used as control input until the next controller is selected. Since the low-level motion controllers take care of dynamic reactions, the high-level switching controller may operate at a low frequency (e. g., 2 Hz). The goal of the high-level controller is to steer the system into a goal state as quickly as possible. The evolution of the belief – and thus also its compressed representation – is stochastic. This results in a stochastic shortest path problem with a continuous, high-dimensional state space, a discrete action space, and constant negative reward. The problem is now handled with the established reinforcement learning machinery. Specifically, we train a *deep Q-Network* (DQN) [24] on off-policy rollouts of the simulated particle filter. The DQN maps the compressed belief $\tilde{b}$ to an index $a^* \in \{1, \ldots, N_{\mathrm{C}}\}$, which selects the currently active motion controller. For an overview of the resulting overall system, we refer to Figure 4.

## V. Implementation & Results

### A. Implementation Details

*Contact Dynamics with Compliance:* We computed the forward dynamics (4) using the MuJoCo physics engine [36] with a constant integrator step-size of 5 ms. As active robot compliance $C$ (see (3)), we used joint-level impedance control with gravity compensation. The corresponding control law is

$$\boldsymbol{\tau}_{\mathrm{r}} = \boldsymbol{M}_{\mathrm{r}}\,(\boldsymbol{q}_{\mathrm{r}})\,\ddot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} + \mathbf{K}\left(\boldsymbol{q}_{\mathrm{r}}^{\mathrm{d}} - \boldsymbol{q}_{\mathrm{r}}\right) + \mathbf{D}\left(\dot{\boldsymbol{q}}_{\mathrm{r}}^{\mathrm{d}} - \dot{\boldsymbol{q}}_{\mathrm{r}}\right) + \boldsymbol{g}_{\mathrm{r}}\,(\boldsymbol{q}_{\mathrm{r}})\,,$$

where $\boldsymbol{M}_{\mathrm{r}}$ and $\boldsymbol{g}_{\mathrm{r}}\,(\boldsymbol{q}_{\mathrm{r}})$ are the inertia matrix and gravitational forces of the robot. Matrices $\mathbf{K} \in \mathbb{R}^{n_{\mathrm{r}} \times n_{\mathrm{r}}}$ and $\mathbf{D} \in \mathbb{R}^{n_{\mathrm{r}} \times n_{\mathrm{r}}}$ denote the positive definite stiffness and damping matrices of the imposed compliance. The stiffness matrix was chosen diagonal with 200 Nm/rad for revolute joints and 800 N/m on all prismatic Degrees of Freedom (DoFs). We chose the matrix $\mathbf{D}$ to achieve critical damping using double-digitalization damping design [3]. For the filter's measurement update (2), we assumed a standard deviation $\sigma_{\mathrm{nom}} = 1.0\,\mathrm{deg}$ on all joints, i. e., an encoder variance $\Sigma_{\mathrm{enc}} = \sigma_{\mathrm{nom}}^2\,\mathbf{I}_{n_{\mathrm{r}}}$.

*Compression:* We obtained the compressed belief from the particle set ($N_{\mathrm{P}} = 300$) using Gaussian mixture expectation-maximization. Attention must be paid since the object configuration $\boldsymbol{q}_{\mathrm{o}}$ is in $\mathrm{SE}\,(3)$ and thus $\boldsymbol{q}$ element of a non-Euclidean vector space. To encode orientations and Euclidean quantities in a common representation, we used the QGMM of Kim et al. [17]. Parallelization and seeding the GMM regression using approximate k-medoids partitioning [25] significantly reduced the fitting time ($\lessapprox 1\,\mathrm{ms}$). Using $N_{\mathrm{G}} = 2$ components has shown to be sufficient for our purposes. To keep the neural net's input space low-dimensional we only used the main diagonal of the covariance matrix for the compressed belief representation. To avoid scaling related instabilities during training, we provided the element-wise logarithm of the covariance as network input.

*Low-Level Controllers:* The low-level controllers operate at 200 Hz. We computed the Jacobian $\mathbf{J}_e$ of the error function and the derivatives of the constraint functions $\boldsymbol{c}_p$ and $\boldsymbol{c}_v$ using an auto-differentiation scheme based on the eTaSL/eTC framework [1]. We chose the proportional gain matrix of the error and constraint dynamics $\mathbf{K}_{\mathrm{p}}$ to obtain a time-constant of 0.2 s. The damping matrix $\mathbf{K}_{\mathrm{d}}$ was chosen so that the dynamics are critically damped. We solved the quadratic program in (7) using qpOASES [9]. Refer to Figure 6 for the number of motion controllers $N_{\mathrm{C}}$ used for the respective task.

*High-Level Controller:* The high-level controller operates at a rate of 2 Hz. To obtain a mapping from the compressed belief to the next action we use a Q-learning update [42] on the compressed belief

$$Q(\tilde{b}, a) \leftarrow Q(\tilde{b}, a) + \alpha \left[ r(\tilde{b},\, a) + \gamma \max_{a'} Q(\tilde{b}',\, a') - Q(\tilde{b}, a) \right],$$

with constant negative reward $r(\tilde{b},\, a) = -1$. We approximate the Q-function using a neural network. Each episode samples a ground-truth candidate from the prior distribution $b_0$. This ground-truth candidate generates the simulated measurement used during the correction step of the particle filter rollouts.

Typical tasks such as in assembly or non-prehensile manipulation require reaching a desired state space target $x_{\mathrm{goal}}$. In such a setting, we terminate an episode as soon as the simulated ground-truth fulfills a goal condition

$$\mathrm{dist}\,(x,\, x_{\mathrm{goal}}) < \epsilon, \tag{8}$$

where $\mathrm{dist}\colon \mathcal{X} \to \mathbb{R}^+$ denotes a suitable distance function and with $\epsilon \in \mathbb{R}^+$ being a distance threshold.

If merely localization is of interest, we define a terminal belief as one with differential entropy

$$-\int_{\mathcal{X}} b_t \log\left(b_t\right) \mathrm{d}x_t < H_{\text{thresh}}. \tag{9}$$

Here, $H_{\text{thresh}} \in \mathbb{R}$ is a given entropy threshold. We approximate the differential entropy of the compressed belief using the upper entropy bound for GMMs given by Huber et al. [15].

The focus of the present paper is not on reinforcement learning. We use it as a tool to create the high-level policy. Therefore, all design choices regarding network architecture and training have been made in a way that minimized debugging and implementation efforts. We strictly separated data generation and policy training using a two-step training scheme. In the first step we generated a data set of 20 000 episodes with purely random actions. The data set was used to train an initial policy. We repeated data generation for another 7 500 episodes, now under purely greedy action selection. The final policy was then trained on the combined data set of random and greedy episodes. Data generation and network training for a single policy took approximately 24 h on a desktop computer (Intel Core i7-7700, 3.6 GHz). We chose the discount factor as $\gamma = 0.98$ and a learning rate of 0.0001. Furthermore, we used the Adam optimizer [18] with $l_2$ kernel regularization (0.01).

Different tasks likely also require networks of different depth. Hence, we make use of the ResNet [13] architecture, which is able to emulate flatter networks if necessary. A first dense layer with 64 outputs is followed by four ResNet blocks. Each ResNet block consists of two dense layers (with each 64 outputs) and ELU activations [7]. To avoid the inherent overestimation of action values in Q-learning we make use of a final Dueling-Layer [41] and double Q-learning [40].

## B. Benchmark Problems

We evaluated the methods using the five benchmark scenarios shown in Figure 6.

*Peg-In-Hole:* Peg-In-Hole is well studied since it represents a large proportion of industrial assembly applications. The present setting consists of a 7-axis redundant robot, which holds a cubical peg. The matching counterpart lies flat on a workspace. It is not fixed in any way. To solve the task, the robot must avoid pushing the workpiece out of its reachable workspace.

*Bridge:* In the BRIDGE benchmark, a 7-axis redundant robot has to push a cube over a narrow passage into a small hole. There is no way to recover if the cube falls off the elevated surface. Improperly selected actions come at a high risk, especially in the early stages of the task. To traverse the narrow bridge, the cube must be well localized.

*Bunny-Gap:* In the BUNNY-GAP task, a 7-axis redundant robot mounted on a linear axis must push a bunny (non-convex, rigid object) across the table through a narrow gap. The bunny fits through the opening lengthways and upright, only. The robot must thus make use of the environment geometry to carefully align the bunny. Early stage actions should be carefully selected to prevent the bunny from being knocked over.

*PR2-Hammer:* This task requires a PR2 robot to lift a hammer from a table surface. The robot must localize the part carefully to avoid the hammer from being pushed off the table. Grasping and lifting is not enabled by means of a single high-level command. The agent must learn the concept of grasping through closing the finger joints and that the gripper needs to remain closed during lifting. In order to enable a grasp, we provide two versions for each of the $n$ target controllers: one that (in addition to accelerating towards the target) opens the gripper and one that closes it.

The above four benchmarks terminate according to (8) with target distances as depicted in Figure 6. The maximum number of actions per episode was chosen as 100.

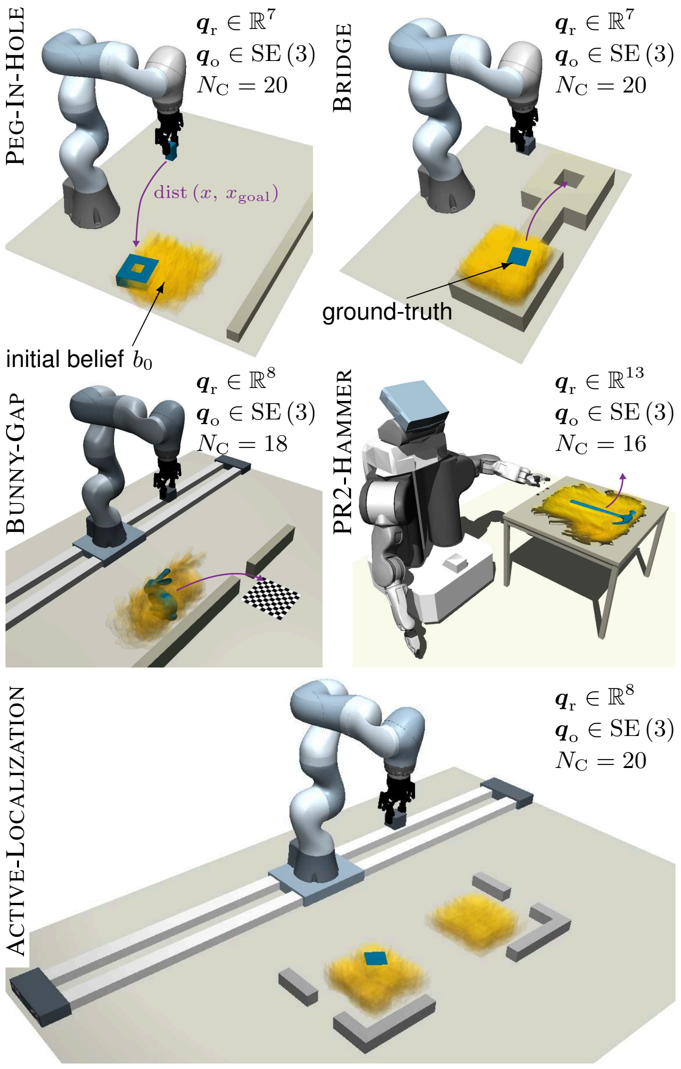


Fig. 6. Benchmark problems from the domains of assembly (PEG-IN-HOLE), non-prehensile pushing (BUNNY-GAP, BRIDGE), grasping (PR2-HAMMER), and active perception (ACTIVE-LOCALIZATION). The yellow area indicates the robot's initial belief. The blue colored object denotes the sampled ground-truth candidate. The robot only uses measurements of the joint position as feedback.

*Active-Localization:* In this task, a cube must be localized. The robot can make active use of the environment geometry to quickly condense the belief, i. e., to reach an entropy goal (9) with target entropy $H_{\text{thresh}} = -8.0\,\text{nat}$. For the active localization task, we abort episodes after 50 actions.

## C. Experimental Setup

To assess the proposed method, we computed policies for each of the benchmark problems. To evaluate individual policies, we performed 100 episode rollouts, with a ground-truth drawn from the respective initial belief $b_0$. Episodes are considered to have failed if the target is not reached within 100 high-level actions (50 s) or if the robot's load limit is exceeded. Due to the present variance, we report the results for 15 networks, each trained on individually generated data sets.

A central question is whether the proposed hierarchical policy actually exploits the uncertainty information encoded in the compressed belief and thereby improves the overall success rate. Put differently: with regard to certainty equivalence, would it be sufficient to separate estimation and control, i. e., to employ a deterministic policy together with the maximum likelihood estimate of a Bayes filter? In this case, we would expect similar results.

To address this, we compared the policies as obtained from the proposed method to a DQN trained on a fully observable environment, that during execution in the partially observable setting, uses a Gaussian maximum-likelihood estimate of the filter's particle set (DQN+MLE).

TABLE I
SUCCESS RATES - TASK BENCHMARKS

| Benchmark | | PEG | BRIDGE | BUNNY | PR2 |
|---|---|---|---|---|---|
| proposed approach | max | 100% | 79.0% | 95.0% | 100% |
| 15 different DQNs | median | **99.0%** | **73.0%** | **90.0%** | **100%** |
| | mean | **98.8%** | **71.2%** | **88.7%** | **98.6%** |
| | min | 97.0% | 46.0% | 74.0% | 85.0% |
| DQN+MLE | max | 67.0% | 75.0% | 93.0% | 60.0% |
| 15 different DQNs | median | 56.0% | 64.0% | 84.0% | 53.0% |
| | mean | 53.9% | 62.2% | 71.9% | 52.4% |
| | min | 30.0% | 51.0% | 29.0% | 42.0% |
| | p-value | $<10^{-2}\%$ | 0.631% | 1.39% | $<10^{-2}\%$ |
| random controllers | | 26.0% | 0% | 1.00% | 8.00% |

## D. Simulation Results

Table I reports the results for the PEG-IN-HOLE, BRIDGE, BUNNY-GAP, and PR2-HAMMER benchmarks. Shown are the success rates under the proposed approach, random controller switching, and the DQN+MLE approach. We state the significance of these results using Welch's two-sided unequal variance t-test, assuming normally distributed success rates.

The proposed approach performed well on all four benchmarks. The BRIDGE task proved to be particularly troublesome whenever the cube was initialized along the edge facing the robot. Due to the robot's kinematic limitations, the robot was then unable to push the object towards the center of the elevated surface.

The proposed approach performs significantly better (p-value $< 5\%$) than DQN+MLE on all tasks. We were able to observe that many of the failure cases of DQN+MLE stem from unrecoverable mistakes during the early localization phase, such as the robot pushing the cube off the elevated surface (BRIDGE) or the hammer off the table (PR2-HAMMER). The proposed method showed less aggressive behaviors in these early stages. Further failure cases of DQN+MLE arose due to its inability to handle multi-modal belief distributions. The formation of multi-modal beliefs was distinctly present on the PEG-IN-HOLE problem. The above results demonstrate that the proposed policy actually exploits the uncertainty information encoded in the compressed belief.

We also evaluated our method's applicability to the task of active localization. Table II shows the success rate (lo-

TABLE II
SUCCESS RATES & COST - ACTIVE PERCEPTION

| filter noise | | $\sigma_{\text{nom}}$ success | $\sigma_{\text{nom}}$ cost | $2\,\sigma_{\text{nom}}$ success | $2\,\sigma_{\text{nom}}$ cost |
|---|---|---|---|---|---|
| proposed approach | max | 90.0% | 48.0 | 87.0% | 49.0 |
| 15 different DQNs | median | **82.0%** | **11.0** | **76.0%** | **15.0** |
| | mean | **79.1%** | **12.9** | **75.0%** | **17.1** |
| | min | 56.0% | 3.00 | 46.0% | 5.00 |
| random controllers | | 77.0% | 23.0 | 51.5% | 34.0 |

calized according to (9) in less than 25 s) and the cost of successful attempts (time spent), under the proposed policy and random interactions, for two levels of measurement uncertainty. Under nominal measurement noise $\Sigma_{\text{enc}} = \sigma_{\text{nom}}^2\,\mathbf{I}_{n_r}$, the filter quickly condensed the belief and both the proposed method and random actions w.r.t. the modes of the distribution performed well at localizing the cuboid. However, 14/15 of our learned policies localized the object significantly (p-value $< 5\%$) faster. With more measurement noise ($\Sigma_{\text{enc}} = (2\,\sigma_{\text{nom}})^2\,\mathbf{I}_{n_r}$), we were able to observe a drop in the success rate when employing random actions w.r.t. the modes of the distribution. The proposed method maintained a high success rate even under the much more conservative measurement update. All of the learned policies performed the task significantly (p-value $< 5\%$) quicker. This demonstrates the proposed method's ability to act purposefully based on the current shape of the belief distribution. We found that for a typical failure case, the policy was unable to condense the belief at all. The entropy remained at the initial level. This happens whenever the upper layer gets stuck, repeatedly selecting non-informative actions. Other failure cases were caused by particle starvation. As a result of too forceful interactions, the particle filter can deplete to a single, highly peaked ($< H_{\text{thresh}}$), yet incorrect hypothesis. To guard the system from such situations, one could introduce a penalty cost for particle starvation during the training phase. This way, a policy can be trained to actively avoid actions that cause impoverished particle sets.

### *E. Real-World Experiment*

As we aim for a model-based approach, we derive our policy only from synthetic data, based on a simulated model of contact dynamics. The policies were executed on real hardware without further processing steps.

Parameters such as surface friction, contact hardness, and models thereof are only a rough approximation. Hence, we expect a gap between the simulated and the real-world environment. To evaluate the impact of this simulation-reality gap, we conducted 60 real-world experiments. Figure 7 shows the experiment platform for the PEG-IN-HOLE benchmark. We used an impedance-controlled KUKA iiwa R800 redundant 7-axis robot. Of the 60 policy executions, 41 succeeded (68.3 %). Many of the failures were caused by particle depletion. The depletion mainly occurred during very forceful interactions, e. g., when the robot scraped the peg along the surface of the hole. During this phase, multiple friction and contact parameters (peg-hole, hole-table) come to play simultaneously and the motion of the real object started to diverge from that of the simulated particles.

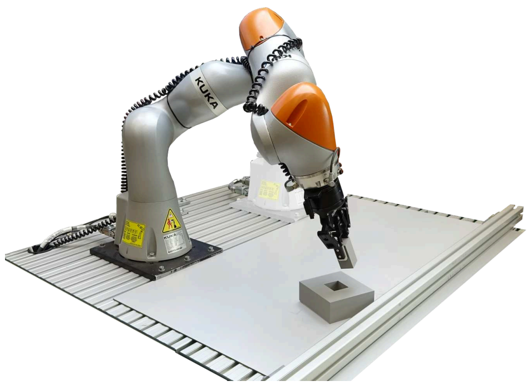


Fig. 7. Real-world experiment platform for the PEG-IN-HOLE benchmark, with an impedance-controlled KUKA iiwa R800 redundant 7-axis manipulator. The clearance between the cubical peg and the hole is 5 mm.

We believe that we can further improve the success rate by enhancing the model's accuracy and by using advanced resampling-schemes that reduce particle starvation. The particle filter could also be extended to provide a continuous estimate of physical coefficients such as friction.

## VI. DISCUSSION

*Conclusion:* In this paper we proposed an integrated system for contact-rich object manipulation under uncertainty. Our method closes the loop from sparse internal sensor feedback to high-frequency, directly executable motion commands. By taking into account the full dynamics, our method is able to consider the kinematic and dynamic limits of the manipulator and can learn to avoid actions associated with a high risk of failure, e. g., dropping or knocking over the object.

We model manipulation under uncertainty as a POMDP and use a particle filter to track the belief. A compressed representation of this belief provides the input to our hierarchical control policy. In this policy, an upper layer situation classifier switches between local motion controllers. These provide the reference for an impedance-controlled robot.

Extensive simulated and real-world experiments based on five different benchmark scenarios demonstrate that the presented approach solves challenging tasks from the domains of active localization, grasping under uncertainty, assembly, and non-prehensile object manipulation.

*Limitations & Future Work:* The proposed approach draws on methods that limit the scope for a more detailed, formal analysis. These include the approximate belief dynamics of the particle filter, belief compression using an insufficient statistic, and a DQN as Q-function approximation. Furthermore, the simulated particle filter rollouts make the generation of the training-data computationally expensive. Finally, the grid of targets for the local motion controllers requires some domain-knowledge. A promising avenue for future work could therefore be to adapt the discrete set of local motion controllers towards a continuous action space for the high-level policy.

## References

[1] E. Aertbeliën and J. De Schutter. eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1540–1546, 2014.

[2] A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato. Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research (IJRR)*, pages 268–304, 2014.

[3] A. Albu-Schäffer, C. Ott, U. Frese, and G. Hirzinger. Cartesian impedance control of redundant robots: recent results with the dlr-light-weight-arms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3704–3709, 2003.

[4] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo. Monte carlo value iteration for continuous-state pomdps. In *Algorithmic Foundations of Robotics IX*, pages 175–191. Springer, 2010.

[5] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 723–730, 2011.

[6] M. Chen, E. Frazzoli, D. Hsu, and W. S. Lee. Pomdp-lite for robust robot planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5427–5433, 2016.

[7] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

[8] M. Egorov. Deep reinforcement learning with pomdps. Technical report, 2015.

[9] H. F. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, pages 327–363, 2014.

[10] N. P. Garg, D. Hsu, and W. S. Lee. Despot-$\alpha$: Online pomdp planning with large state and observation spaces. In *Robotics: Science and Systems (RSS)*, 2019.

[11] K. Hauser. Randomized belief-space replanning in partially-observable continuous spaces. In *Algorithmic Foundations of Robotics IX*, pages 193–209. Springer, 2010.

[12] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI Fall Symposium Series*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Grasping pomdps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4685–4692, 2007.

[15] M. F. Huber, T. Bailey, H. Durrant-Whyte, and W. D. Hanebeck. On entropy approximation for gaussian mixture random vectors. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 181–188, 2008.

[16] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research (IJRR)*, pages 1194–1227, 2013.

[17] S. Kim, R. Haschke, and R. Ritter. Gaussian mixture model for 3-dof orientations. *Robotics and Autonomous Systems*, pages 28–37, 2017.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[19] M. C. Koval, D. Hsu, N. S. Pollard, and S. S. Srinivasa. Configuration lattices for planar contact manipulation under uncertainty. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[20] M. C. Koval, N. S. Pollard, and S. S. Srinivasa. Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty. *The International Journal of Robotics Research (IJRR)*, pages 244–264, 2016.

[21] H. Kurniawati and V. Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *International Symposium on Robotics Research (ISRR)*, pages 611–629, 2013.

[22] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems (RSS)*, 2008.

[23] T. P. Le, N. A. Vien, and T. Chung. A deep hierarchical reinforcement learning algorithm in partially observable markov decision processes. *IEEE Access*, pages 49089–49102, 2018.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, pages 529–533, 2015.

[25] R. T. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge & Data Engineering*, pages 1003–1016, 2002.

[26] C. Phiquepal and M. Toussaint. Combined task and motion planning under partial observability: An optimization-based approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9000–9006, 2019.

[27] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1025–1032, 2003.

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

*(RSS)*, 2010.

[29] R. Platt, L. P. Kaelbling, T. Lozano-Pérez, and R. Tedrake. Efficient planning in non-gaussian belief spaces and its application to robot grasping. In *International Symposium on Robotics Research (ISRR)*, pages 253–269. Springer, 2017.

[30] J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart. Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research (JMLR)*, pages 2329–2367, 2006.

[31] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research (IJRR)*, pages 1448–1465, 2009.

[32] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. pages 35–40, 1999.

[33] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. v. Wichert, and W. Burgard. Modeling and planning manipulation in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 176–182, 2019.

[34] K. M. Seiler, H. Kurniawati, and S. P. N. Singh. An online and approximate solver for pomdps with continuous action space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2290–2297, 2015.

[35] S. Thrun. Monte carlo pomdps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1064–1070, 2000.

[36] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

[37] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Physical Review*, pages 823–841, 1930.

[38] J. Van Den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research (IJRR)*, pages 895–913, 2011.

[39] 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 (IJRR)*, pages 1263–1278, 2012.

[40] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

[41] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1995–2003, 2016.

[42] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, pages 279–292, 1992.

[43] F. Wirnshofer, P. S. Schmitt, P. Meister, G. v. Wichert, and W. Burgard. State estimation in contact-rich manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3790–3796, 2019.