Belief-Space Planning using Learned Models with Application to Underactuated Hands

Andrew Kimmel*, Avishai Sintov*, Juntao Tan, Bowen Wen,
Abdeslam Boularias, Kostas E. Bekris

Abstract Acquiring a precise model is a challenging task for many important robotic tasks and systems - including in-hand manipulation using underactuated, adaptive hands. Learning stochastic, data-driven models is a promising alternative as they provide not only a way to propagate forward the system dynamics, but also express the uncertainty present in the collected data. Therefore, such models enable planning in the space of state distributions, i.e., in the belief space. This paper proposes a planning framework that employs stochastic, learned models, which express a distribution of states as a set of particles. The integration achieves anytime behavior in terms of returning paths of increasing quality under constraints for the probability of success to achieve a goal. The focus of this effort is on pushing the efficiency of the overall methodology despite the notorious computational hardness of belief-space planning. Experiments show that the proposed framework enables reaching a desired goal with higher success rate compared to alternatives in simple benchmarks. This work also provides an application to the motivating domain of in-hand manipulation with underactuated, adaptive hands, both in the case of physically-simulated experiments as well as demonstrations with a real hand.

1 Introduction

Uncertainty during fabrication, underactuation, soft contacts and partial observability are frequent reasons that makex an analytical model infeasible for many important robotic tasks and systems. For example, 3D-printed, underactuated compliant hands [24] differ in size, weight, inertia, stiffness and friction depending on the manufacturing technique. Due to the resulting uncertainties, manually designing precise models for these hands is a significant challenge on top of the difficulty in modeling passively elastic joints [13, 36]. The lack of accurate analytical models complicates the control and planning for such systems, especially for tasks that are dynamic in nature, such as within-hand manipulation.

A popular solution to this limitation is to acquire a stochastic model based on recorded transition data, such as through the application of a Gaussian Process (GP) [33]. Given the current state and a desired action, the stochastic model provides a distribution of the next state exhibiting the uncertainty in the data. Then, this allows

The authors are with the Computer Science Department at Rutgers University and their work was supported by NSF Awards 1734492 and 1723869.
*Equal contribution.
Corresponding authors: andrew.kimmel@cs.rutgers.edu, avishai.sintov@rutgers.edu.

a motion planner to use the stochastic model to propagate forward the system's dynamics. A naïve approach would be to sample from the output distribution of the stochastic model to acquire the next state. This, however, eliminates reasoning about the current belief, i.e., the distribution of possible states. Therefore, the roll-out of the plan has high likelihood of deviating from the desired path and fail.

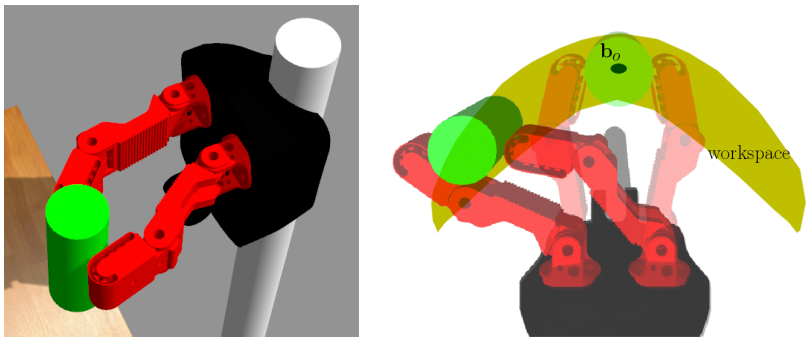

Fig. 1 Physics engine simulation of the Model T42 adaptive hand in the Gazebo environment (left). A visualization of the approximated workspace (based on data) of the hand, shown in yellow, along with the initial belief distribution $\mathbf{b}_o$ (right).

This paper proposes a belief-space planning framework, which utilizes stochastic, data-driven models that represent beliefs as a collection of particles [29]. The algorithm leverages the uncertainty expressed by the learned stochastic model to find a path that guarantees a minimum probability of success given the model and which can be optimized over time. The quality of a path can be measured by metrics, such as duration of execution, while the probability of success is influenced by the potential interaction of a path with invalid regions, such as obstacles or dropping an object during within hand manipulation, given the belief distribution.

Planning in belief space with learned models is equivalent to planning for systems with complex dynamics where there is no steering function and can only forward propagate the dynamics. In prior work, we proposed one of the first sampling-based planners, which achieves asymptotic optimality for systems with dynamics [19, 20]. We then extended it to anytime belief space planning for a Non-Observable Markov Decision Process (NOMDP) given particle set representations of beliefs, where a critical choice is the distance function in the belief space [23].

The current work describes how to incorporate learned stochastic models in sampling-based belief-space planning. This objective is to achieve increasingly better paths as a function of computation time, which also guarantee a certain level of robustness for challenging tasks that are difficult to model analytically. Furthermore, the work improves the practical computational efficiency of the integrated framework. The paper provides an evaluation through a set of simple systems followed by an application on in-hand manipulation with underactuated adaptive hands, including both in a physics-engine and demonstrations on the real hand. Two different stochastic data-driven models for adaptive hands were used in this process, one based on prior recent work [36], which utilizes Gaussian processes, and a new data-driven model based on Bayesian Neural Networks.

2 Related Work

Belief Space Planning Interestingly, some promising efforts in belief-space planning uses similar tools to those used in the deterministic case. In particular, sampling a small set of representative beliefs and planning with respect to only this small set of sampled beliefs. In fact, many methods (e.g., [2, 29, 32]) in belief-space planning are extensions of sampling-based ones for the deterministic case, such as PRM, RRT, PRM*, and RRG [5, 15]. These methods typically restrict beliefs to be represented by Gaussian parameters or consider the maximum likelihood estimate of the state.

Recent work, which has shown that one can achieve asymptotic optimality without a steering function in the deterministic case [19, 22], has the potential to allow an even more straightforward way to extend sampling-based planners to belief-space planning. The similarity between deterministic planning without a steering function and belief-space planning indicates that properties critical for deterministic motion planning are likely to be critical for belief-space motion planning as well.

Similar to sampling-based methods for the deterministic case, many equivalent approaches for belief-space planning rely on distances between beliefs to partially guide their sampling and pruning operations [16, 17, 38]. Many distance functions can be potentially used and they can have significantly different effects in belief-space planning. Nevertheless, the effectiveness of the different distance functions has not been studied in the related literature on belief-space planning.

Planning and Control with Learned Stochastic Models Planning and control using data-driven transition models is becoming increasingly a popular alternative to model-free methods [21], especially in low-dimensional problems where data efficiency is critical [27, 28, 18, 8, 21, 6], and to analytical models that are difficult to accurately handcraft in practice. Prior works on trajectory search methods using learned stochastic models often use a Gaussian Process [28, 8] or a Bayesian neural network [27] to represent the transition function and learn it from data. The belief state is approximated with a Gaussian by using the moment matching technique, and propagated in time for forecasting future trajectories and optimizing a parameterized controller. The Gaussian approximation is required for deriving a gradient-based optimization of the trajectory [8]. This requirement was relieved in other works where the belief state is approximated by a set of particles [27]. One study [27] has shown that moment matching is a crucial limitation, but did not present a planning or control method that utilizes the belief state particles. Another thorough study [6] also clearly demonstrated the benefit of particles over moment matching. Trajectories in [6] are however obtained from the generic black-box cross-entropy method (CEM), which is computationally expensive [1]. Learned, stochastic, transition models are also gaining in popularity in high-dimensional problems, as shown by the recent works on deep visual foresight for planning robot motion [14, 9, 39, 35, 31]. Particle propagation are the method of choice in these applications, although there has not been any work on robust planning in these applications to the best of our knowledge. This work differs from the aforementioned works mainly by considering robustness constraints in a belief space planner to deal with uncertainties that are inherent to data-driven models.

3 Problem Definition

Let $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^n$ be an observable state vector of a given system and $\mathbf{u} \in \mathbb{U}$ be an action taken from a set $\mathbb{U}$ of possible actions. The state space is decomposed into a valid subsect $\mathbb{X}_{valid}$ and an invalid one. Validity typically refers to the state being collision-free but it can also consider other types of failure regions. The system is governed by the unknown transition $f : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$, such that a given state-action pair $(\mathbf{x}_t, \mathbf{u}_t)$ at time t is mapped to the next state $\mathbf{x}_{t+1}$ according to $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. Denote as $\mathbf{b}_t$ the belief distribution over the state space $\mathbb{X}$ at time t. The initial belief is assumed to be unimodal, for instance, distributed according to a Gaussian $\mathbf{b}_o \sim \mathbb{N}(\mu_o, \Sigma_o)$.

This work considers a Non-Observable Markov Decision Process (NOMDP) problem [23]. In this setup, observations are not available and planning is performed solely based on the initial belief $\mathbf{b}_o$. The problem is to plan a path π, which reaches a region $\mathbb{G}(\mathbf{x}_g)$ for the goal state $\mathbf{x}_g$ with high probability, which involves the following two aspects:

a) the minimum probability of the path being valid should be above a threshold δ, i.e., $Prob(\pi(t) \in \mathbb{X}_{valid}) > \delta \; \forall \; t \in [0 : T]$, where $\pi(t)$ is the state along the path π of duration T at time t, and
b) the probability of successfully reaching the goal region $\mathbb{G}$ must be above a threshold κ, $Prob(\pi(T) \in \mathbb{G}(\mathbf{x}_g)) > k$, where $\pi(T)$ is the final state along path π of duration T.

Given these constraints, the path should be optimizing a cost function C, such as path duration or length. The next section discusses the learning of a stochastic model, followed by the planning framework addressing the problem defined here.

4 Learning a model

As discussed previously, a precise model of a system $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ is not always available. An approximate data-based model $\mathbf{x}_{t+1} = \tilde{f}(\mathbf{x}_t, \mathbf{u}_t)$ offers an alternative and can be used in model-based algorithms. This section briefly describes stochastic data-based models that encapsulate both state uncertainty and uncertainties in the training data. These models are applicable to the motivating domain of within-hand manipulation and are used in the experimental section.

A training set is acquired by applying random actions to the system, while recording the observable states. Consequently, the resulting data is a set of state-action trajectories $\mathbb{P} = \{\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_k\}$, where $\bar{\mathbf{x}}_i = (\mathbf{x}_i^T, \; \mathbf{u}_i^T)^T$. Later, the trajectories in $\mathbb{P}$ are pre-processed to a set of training inputs $\bar{\mathbf{x}}_i$ and corresponding output labels of the next state $\mathbf{x}_{i+1}$ to define the training set $\mathbb{T} = \{(\bar{\mathbf{x}}_i, \mathbf{x}_{i+1})\}_{i=1}^N$.

For each $\bar{\mathbf{x}}_i$, we also label $\mathbf{d}_i = \{\texttt{success}, \texttt{fail}\}$ indicating whether the transition from $\mathbf{x}_i$ with action $\mathbf{u}_i$ resulted in a failure. In the context of within-hand manipulation, a failure corresponds to the hand dropping the manipulated object. A classifier, such as a neural network or a Support Vector Machine (SVM), is trained from the labeled data in terms of failures. The classifier provides the failure probability $Prob(\mathbf{x}_{t+1} \notin \mathbb{X}_{valid}) \in [0, 1]$ for the state x_{t+1} that arises from the application of action $\mathbf{u}_t$ at a given state $\mathbf{x}_t$.

4.1 Stochastic Data-based Models

Given the training data $\mathbb{T}$, for any state-action pair $\bar{\mathbf{x}}_t$ we aim to learn the distribution of the next state $p(\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t) = p(\tilde{f}(\mathbf{x}_t)|\bar{\mathbf{x}}_t)$. Next, we briefly discuss two data-based methods for this stochastic regression problem. We demonstrate the use of these methods in the evaluation section.

4.1.1 Gaussian Processes

A model can be learned by using a Gaussian Process (GP) [33], which is a non-parametric regression method. Let k be a kernel function, $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, for measuring the similarity between two states in $\mathbb{R}^n$. Let $X_N = \left(\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_N\right)$ be a set of observable hand-object states, and let $Y_N = \left(\mathbf{y}_1, \mathbf{y}_2 \ldots, \mathbf{y}_N\right)$ be the set of observable hand-object states resulting from applying a given action in the states in X_N, where $\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathbb{N}(0, \sigma^2)$ is independent white noise. The kernel *Gram matrix* K is an $N \times N$ positive-definite matrix, defined as $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, for $i, j \in \{1, \ldots, N\}$. Given X_N and Y_N, the posterior distribution on the values of f in any state $\mathbf{x}^*$ is a Gaussian with mean vector $\mu(\mathbf{x}^*)$ and covariance matrix $\Sigma(\mathbf{x}^*)$,

$$\mu(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T \beta, \tag{1}$$

$$\Sigma(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}^*), \tag{2}$$

where $\beta = (K + \sigma^2 I)^{-1} Y_N$, $\mathbf{k}(\mathbf{x}^*) = k(X_N, \mathbf{x}^*)$, and I is an identity matrix.

The computational bottleneck in evaluating the GP is computing the Cholesky decomposition of the covariance function. Thus, the computational complexity scales cubically $O(N^3)$ with the size of the dataset N, making often global regression infeasible [30]. Nearest-neighbor GP provides a scalable alternative by using local information for regression [7]. Only data points in the proximity of the query point are used to make a prediction [36]. For each query state-action pair, a set of nearest-neighbors is searched and used for regression.

4.1.2 Bayesian Neural Networks

While a conventional Artificial Neural-Network (ANN) can provide global regression, its output is deterministic and cannot reason about confidence in the result. That is, an ANN does not output model uncertainty. Bayesian Neural Networks (BNN) [26], on the other hand, provide model uncertainty by incorporating posterior distributions over the weights of the network. The use of dropouts in ANN yields a Bayesian approximation of the uncertainty [10], similar to GP's.

5 Robust Planning with Learned Models

In this section we describe the core components of the framework: propagating the belief state and planning over belief space.

5.1 Belief-State Propagation

Given the current state $\mathbf{x}_t$ and action $\mathbf{u}_t$, the learned models described in Section 4 are able to provide a distribution for the next state $p(\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t)$. Uncertainty about the current state would, however, requires integrating the predictive distribution over the

current distribution. That is, given a distribution for the current state-action[1] $p(\bar{\mathbf{x}}_t)$, the distribution of the next state is given by the integration of

$$p(\mathbf{x}_{t+1}) = \int_{\bar{\mathbf{x}}_t} p(\bar{\mathbf{x}}_t) p(\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t) d\bar{\mathbf{x}}_t. \tag{3}$$

Distribution $p(\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t)$ is given by the posterior distribution and is a non-linear function. Therefore, the belief for the next state $p(\mathbf{x}_{t+1})$ is not Gaussian and its integral is analytically intractable [12]. For that matter, we approximate the distribution of the next state by propagating particles through the model. Particles propagation was proposed in [29] to propagate uncertainty. A set of particles is stored in each node to represent the corresponding distribution. Then, propagation from a node is performed by simulating each particle along with a desired action, to acquire a new set of particles that approximates the distribution of the next node. Similarly, in [12], a Monte-Carlo method was used to approximate a Gaussian for the propagated belief. Samples from the current belief are propagated through a GP yielding a new set of propagated samples. The mean and variance can then be taken to represent the next belief state. In this work, we combine these two notions to approximate belief states through a stochastic data-based model with Particles Propagation (PP). A set of M particles $\mathbb{B}_t$ from distribution $\mathbf{b}_t \sim p(\bar{\mathbf{x}}_t)$ is propagated through the model to acquire a new set $\mathbb{B}_{t+1}$. We denote this mapping as

$$\mathbf{b}_{t+1} = \Gamma(\mathbf{b}_t, \mathbf{u}_t) \tag{4}$$

(or $\mathbb{B}_{t+1} = \Gamma(\mathbb{B}_t, \mathbf{u}_t)$). Prior to propagation, all particles are checked for validity through the classifier, using p_s (for each $\mathbf{x}_t^i \in \mathbb{B}_t$). The invalid particles are counted in M_f. To preserve constant number of particles, failed ones are omitted and replaced by duplicating survived particles in $\mathbb{B}_t$. Furthermore, the probability of $\mathbf{b}_{t+1} \approx \mathbb{B}_{t+1}$ being valid is

$$p_{valid}(\mathbf{b}_{t+1}) = \left(1 - \frac{M_f}{M}\right) \tag{5}$$

The process of propagating a set of particles is described in Algorithm 1.

Note that in the case of a GP model, $GP(\mathbb{B}_t, \mathbf{u}_t)$ is a batch particles propagation (BPP) version of (1)-(2) where the states of all particles in $\mathbb{B}_t$ are concatenated before applied, providing a much faster computation than applying (1)-(2) for each particle individually. We note that an analytical solution exists that provides a Gaussian approximation of (3) using exact moments [8, 12]. However, the computational efficiency of the method was tested to be rather low compared to the BPP with GP and did not prove to be better in accuracy. Thus, we do not include the method in this paper.

[1] Although actions are certain in some of our test cases, for simplicity and generality, we denote actions as uncertain along with the state.

Algorithm 1: PP $(\mathbb{B}_t, \mathbf{u}_t)$

Initialize empty set $\mathbb{B}_{t+1}$;
Init $M_f \leftarrow 0$;
foreach $\mathbf{x}_t^i \in \mathbb{B}_t$ **do**
  if $random([0,1]) > p_s(\mathbf{x}_t^i, \mathbf{u}_t^i)$ **then**
    Remove $\mathbf{x}_t^i$ from $\mathbb{B}_t$;
    $M_f \leftarrow M_f + 1$;
  end
end
$p_{valid}(\mathbf{b}_{t+1}) \leftarrow \left(1 - \frac{M_f}{|\mathbb{B}_t|}\right)$;
Sample M_f points from $\mathbb{B}_t$ and add to $\mathbb{B}_t$; `// Duplicates valid particles to keep` $|\mathbb{B}_t|$ `consistent`
$\mathbb{B}_{t+1} \leftarrow \Gamma(\mathbb{B}_t, \mathbf{u}_t)$;
return $\mathbb{B}_{t+1}$, $p_{valid}(\mathbf{b}_{t+1})$;

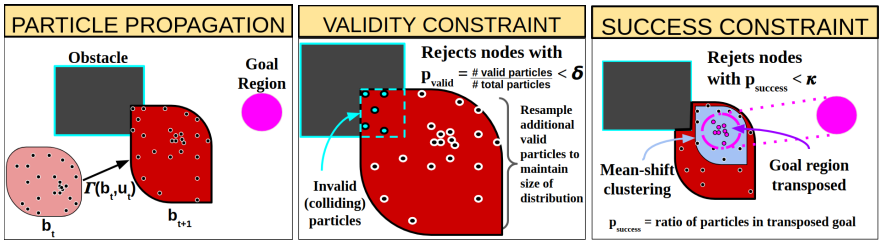

Fig. 2 The uncertainty of the system is captured using a particle distribution over the system's state. Using the black-box propagation model, each particle is propagated to obtain the next state's distribution (left). In order for the new node to be added to the tree, two robustness constraints are checked. First, the ratio of valid particles over the total must exceed the threshold δ (middle). Validity can refer to constraints such as being collision-free, or other criteria, depending on the system. Second, the most dense cluster of particles is computed through mean-shift, and the goal region is transposed to this center (right). The ratio of particles that lie within this transposed region must exceed the threshold κ.

5.2 Belief-Space Planning

The objective of the planner is to a compute a sequence of controls $\mathbf{u}^*$ for a system affected by noise, in the state and/or control space, such that the trajectories rolled out from these controls have a high likelihood of reaching the goal and remaining valid. There are two main components necessary to perform such kinodynamic motion planning - state validity and state transition. Related work has proposed using learned models built from collecting data from the adaptive hand to generate a classification of the valid state space [4] and a state transition model [36]. Given as input the belief of the current state $\mathbf{b}_t$, and the desired control to apply $\mathbf{u}_t$, the transition model (4) provides the belief of the next state, while the validity model provides the probability of being valid $\mathbf{P}_{valid}(\mathbf{b}_{t+1})$.

For deterministic planning, the probability of success can be treated in a binary fashion (i.e. setting a strict threshold of 50%), allowing for a classification of the invalid portion of the state space. Equipped with these tools, it is possible to apply a standard search-tree method (e.g. an A*-like) attempting to solve this problem. There are several issues, however, which make such a search process intractable.

Namely, the high dimensionality of the system prevents easy discretization of the state space. Additionally, recall that an A* will expand the states of all neighbors first, thereby requiring several calls to the transition model at each iteration. However, the propagation time of a set of particles is non-trivial, so avoiding this as much as possible can greatly speed up the search.

Algorithm 2: ROBUST $(\mathbf{b}_o, \mathbb{G}(\mathbf{x}_g), \kappa, \delta, \lceil t \rceil)$

1 Initialize probabilities: $p_{valid}(\mathbf{b}_o), p_{success}(\mathbf{b}_o) \leftarrow 1.0$;
2 Initialize belief-space tree: $\mathbb{T} \leftarrow \{\mathbf{b}_o, p_{valid}(\mathbf{b}_o), p_{success}(\mathbf{b}_o)\}$;
3 Initialize candidate action set: $\mathbb{U}_{cand}(\mathbf{b}_{new}) \leftarrow NULL$;
4 **while** $t \leq \lceil t \rceil$ **do**
5 **if** $\mathbf{b}_{t-1} \neq \emptyset$ **and** $h(\mathbf{b}_{t-1}) < h(\mathbf{b}_{t-2})$ **then**
6 $\mathbf{b}_{sel} \leftarrow \mathbf{b}_{t-1}$; // bias selecting node which improved heuristic
7 **else**
8 $\mathbf{b}_{sel} \leftarrow$ SearchSelection() ; // sample state space to select node
9 $\mathbb{U}_{cand}(\mathbf{b}_{sel}) \leftarrow$ GenerateCandidateActions$(\mathbf{b}_{sel}, \delta)$;
10 $\mathbf{u}_{best} \leftarrow \underset{\mathbf{u} \in \mathbb{U}_{cand}}{argmin}\, h(\mathbf{b}_{sel}, \mathbf{u})$; // select action with best predicted heuristic
11 $\mathbb{U}_{cand}(\mathbf{b}_{sel}) \leftarrow \mathbb{U}_{cand}(\mathbf{b}_{sel}) \setminus \mathbf{u}_{best}$; // remove selected action from action set
12 $\mathbf{b}_t, p_{valid} \leftarrow$ PP$(\mathbf{b}_{sel}, \mathbf{u}_{best})$; // Alg. 1 generates next belief
13 $cost(\mathbf{b}_t) \leftarrow cost(\mathbf{b}_{sel}) + cost(\mathbf{u}_{best})$; // compute the cost of executing $\mathbf{u}_{best}$
14 $p_{valid}(\mathbf{b}_t) \leftarrow \min(p_{valid}(\mathbf{b}_{sel}), p_{valid})$; // retain the minimum p_{valid}
15 $p_{success} \leftarrow$ ComputeOverlap$(\mathbf{b}_t, \mathbb{G}(\mathbf{x}_g))$; // compute success probability
16 $p_{success}(\mathbf{b}_t) \leftarrow \min(p_{success}(\mathbf{b}_{sel}), p_{success})$; // retain the minimum $p_{success}$
17 **if** $cost(\mathbf{b}_t) > cost(\mathbf{u}^*)$ **or** $p_{valid}(\mathbf{b}_t) < \delta$ **or** $p_{success}(\mathbf{b}_t) < \kappa$ **then**
18 $\mathbf{b}_t \leftarrow \emptyset$; // prune node which does not meet the constraints
19 **if** $\mathbf{b}_t \neq \emptyset$ **then**
20 $\mathbb{T}, \mathbf{u}^* \leftarrow$ AddEdge$(\mathbf{b}_{sel} \rightarrow \mathbf{b}_t, p_{valid}(\mathbf{b}_t), p_{success}(\mathbf{b}_t))$; // add new edge to tree
21 $\mathbb{U}_{cand}(\mathbf{b}_t) \leftarrow NULL$;
22 **return** $\mathbf{u}^*$; // the best found action sequence within the time limit

The high level-algorithm for the planing process is presented in Alg. 2, which is inspired by a search strategy capable of using heuristics [3, 22]. Given an initial belief $\mathbf{b}_o$, a desired goal region $\mathbb{G}(\mathbf{x}_g)$ (defined as hyper-ball around $\mathbf{x}_g$), the validity constraint threshold κ, the success constraint threshold δ, and the planning time limit $\lceil t \rceil$, the planner must return a solution action sequence $\mathbf{u}^*$ which optimizes some cost function (i.e. path length). This planning process ensures the following robustness constraints on trajectories rolled out using $\mathbf{u}^*$. First, the probability of the trajectory remaining valid throughout its execution is above the threshold δ. This is computed by evaluating the ratio of particles which are valid at each step of the trajectory. Second, the probability of the trajectory reaching the goal region is above the threshold κ. This is maintained by following the uncertainty pruning condition, and the final goal check verifies that the ratio of particles within the goal region is above the threshold (Fig. 2 right).

Selecting Nodes for Expansion: Each iteration of the algorithm starts by selecting a node of the belief-space tree so as to expand it (lines 5-8). If a node was added during the previous iteration, and its heuristic value is better than its parent on the tree, then the newly added node is selected for expansion (line 5-6). The heuristic

value $h(\mathbf{b})$ of a belief-state corresponds to an optimistic estimate of the cost to reach $\mathbb{G}(\mathbf{x}_g)$. Otherwise, if there was no progress made in the previous iteration towards reaching $\mathbb{G}(\mathbf{x}_g)$, the `SearchSelection` subroutine (line 8) instead uses a probabilistic selection process, where the probability of selecting a node depends on the node's corresponding sum of cost from the root and heuristic cost to reach the goal. All nodes are guaranteed to have a non-zero probability of being selected; however, nodes with better costs and heuristic sum have a greater probability.

Ordering Candidate Actions: `ROBUST` iterates over the generated actions, which are prioritized in terms of the lowest h, and the best action $\mathbf{u}_{best}$is considered for addition at each iteration (line 11). The standard approach to ordering the candidate action set $\mathbb{U}_{cand}$ for a given search tree node is to prioritize actions with better heuristic. Doing so allows for promising actions to be executed first, reducing the need to spend expensive propagation steps on every action out of the node. The choice of heuristic could significantly improve the performance of the planner. However, since this is not the focus of this paper, we have elected to instead use the standard Euclidean distance in state space as the heuristic in each benchmark.

Particle Propagation: Once a node and action is selected, the algorithm propagates to find the next belief state (line 12), the total cost incurred (line 13), and probability of being valid (line 14). Rather than keep track of cumulative probabilities, our method imposes that the bottle-neck (i.e. minimum) probability is constrained. Otherwise, for longer sequences of controls, these probabilities can quickly approach an infinitesimal number.

Pruning Conditions: The probability of successfully reaching the goal (lines 15-16) is also computed, which is accomplished by applying mean-shift to the belief distribution of the current state, computing the highest density region of particles (Figure 2 (right)). Because of the NOMDP nature of the problem, the uncertainty of state (i.e. its variance) can only increase over time. Hence, the $p_{success}$ can be treated as a pruning condition. A node is pruned from the tree if: a) it's cost incurred is greater than the best solution cost found; b) the $p_{valid} < \delta$; or c) the $p_{success} < \kappa$. If the node $\mathbf{b}_t$ passes these checks (line 17), then it is added to the tree $\mathbb{T}$ (line 20). Furthermore, if a better path to the goal is discovered with the addition of $\mathbf{b}_t$, it is recorded as $\mathbf{u}^*$ (line 20).

6 Experimental Evaluation

We evaluate the proposed method over a variety of noisy systems. Initially, the framework is evaluated independently of the data-driven learned models, to show the efficacy of the proposed planner. Next, we build either a GP or a BNN over a point system, a physics-engine simulated adaptive hand, and the real adaptive hand. Experiments are comprised of a planning phase, which computes a trajectory to execute, and a rollout phase, which executes the trajectory several times.

6.1 Planning results

We experimentally evaluate the validity of the proposed approach, as well as several comparison points, across multiple different systems, both in simulation and

on the real robot. Unless otherwise specified in the experiment, the following algorithms, experiment setup, and metrics were used in the evaluation.

Algorithms: ROBUST is the proposed belief-space planning approach, MEAN-ONLY is similar to the proposed approach but reinitializes all particles to be the mean at each time-step, and STANDARD is a deterministic approach (i.e. no particles). All methods make use of a learned stochastic transition model and a failure classifier (both obtained from data), as discussed in Section 4.

Setup: All methods were evaluated on a single Intel Xeon E5-4650 processor with 8 GB of RAM. First, for the 2D Point System, Simulated Hand, and Real Hand, data was collected to generate the transition model and classifier. Next, the planning approaches were given these models and tasked with computing a solution for reaching a goal region within a specified time limit. Solutions that were found within this time limit were subsequently rolled out multiple times.

Metrics: In each experiment (unless otherwise stated), we evaluate each planning approach in terms of initial solution and planning solved rate (solution returned within a specified time limit). Since the planner is asymptotically-optimal, we report both the initial and final solution path lengths (cumulative state space distance). Solutions found within the time limit are rolled out several times, and we report both the ratio of paths that reached the goal (reached goal rate) and the ratio of paths that remained valid (validity rate).

6.1.1 Noisy Control Space Example

This experiment evaluates the proposed planning framework for a 2-link acrobot, which is passive-active [37] and has a 4 dimensional state space. The system is controlled through torque on the second joint. Although the system is low dimensional, it is highly non-linear. Additionally, we introduce noise into the controls in the following fashion: torques which are within 50% of the control limits have zero noise, while torques above this threshold have Gaussian noise added to it. The system is tasked with reaching an upright balancing position with near-zero velocity while avoiding obstacles, including self collisions. Actions for the system are generated by sampling in the continuous control space. The heuristic is the distance between the given state and the upright goal in task space. We evaluate both STANDARD and ROBUST, with a planning time limit of 5 minutes, and constraint thresholds $\{\delta := 0.8, \kappa := 0.7\}$. Figure 3 shows an example of a planned path and multiple rollouts for each algorithm. Quantitative results are reported in Table 1.

Table 1 Results for the Noisy Acrobot system averaged over 25 different trials.

Acrobot Experiment

	STANDARD	ROBUST
Initial Solution Time [s]	221.5	312.5
Initial Solution Path Length	52.4	72.7
Final Solution Path Length	47.8	59.3
Planning Solved Rate	0.68	0.64
Reached Goal Success Rate	0.00	0.72
Validity Rate	0.16	0.84

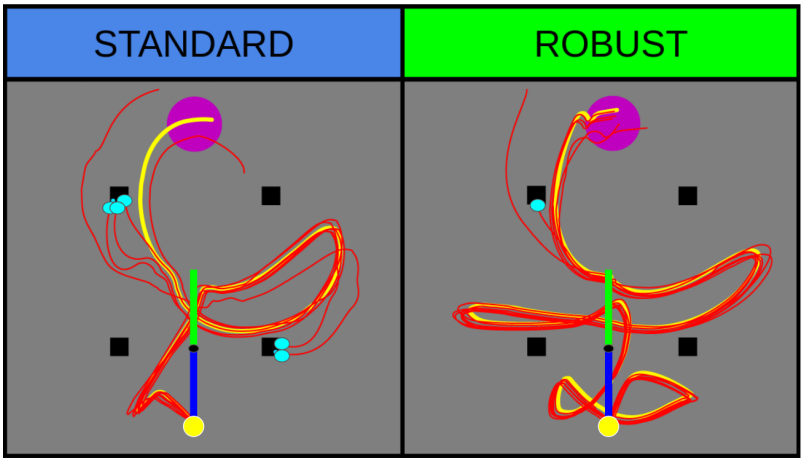

Fig. 3 Example rollouts for the acrobot showing the planned path (yellow) vs. multiple rollouts (red) for the `STANDARD`(left) and `ROBUST`(right) algorithms. Paths which collide are marked with a blue dot at the collision point.

Remarks: As shown in Table 1, `STANDARD` had the smallest computation time and consequently the highest planning solved rate. Given that `STANDARD` does not maintain a belief-state during planning, these results are expected. `STANDARD` also had the shortest initial and final solution path lengths, indicating that it was using high-torque controls to quickly steer the system to the goal state. However, rolling out these actions showed that the `STANDARD` solutions were incapable of reaching the goal (0% reached the goal) and often collided (only 16% were collision-free and valid). In contrast, `ROBUST` had longer computation time (via the particle propagation) and consequently a lower planning solved rate. Furthermore, both the initial and final solution paths from `ROBUST` were longer. However, rolling out the `ROBUST` solutions resulted in 72% of the paths reaching the goal region, and 84% of paths remained valid. These results indicate that the proposed planning framework is successfully satisfying the imposed robustness constraints.

6.1.2 Noisy State Space Experiment

This experiment evaluates the proposed planning framework for a 2D Point system with a data-based learned model, and where noise is introduced into different regions of the state space. Each "corridor" in the state space corresponds to a specific combination of state space noise and probability of being valid p_{valid}. Referring to Figure 4, these scenarios are (from bottom corridor to top): {high noise, low p_{valid}},{high noise, high p_{valid}},{low noise, low p_{valid}}, and {low noise, high p_{valid}}. For this setup, approximately $500,000$ transition points were collected while applying random discrete actions corresponding to the eight cardinal directions. With the collected data, we trained a GP local regressor of 100 nearest neighbors.

Remarks: As shown in Table 2, `STANDARD` again has the smallest computation time. Both `STANDARD` and `MEAN-ONLY` were unable to avoid the {high noise, high p_{valid}} corridor in the state space, and consequently the rollouts of their solutions

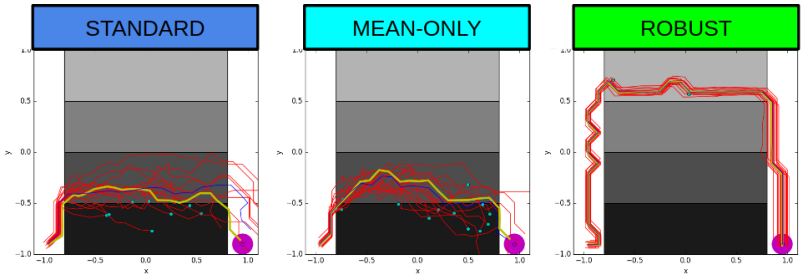

Fig. 4 Experimental results for the point-system showing plots of the planned path vs. multiple rollouts for the `STANDARD` (left), `MEAN-ONLY` (middle), and `ROBUST` (right) algorithms.

Table 2 Results for the 2D Point System averaged over 25 different trials.

2D Point Experiment

	`STANDARD`	`MEAN-ONLY`	`ROBUST`
Initial Solution Time [s]	2.9	25.1	61.7
Initial Solution Path Length [mm]	29.2	28.8	61.2
Final Solution Path Length [mm]	28.2	27.5	54.0
Planning Solved Rate	1.0	1.0	1.0
Reached Goal Success Rate	0.04	0.08	.96
Validity Rate	0.24	0.2	.96

resulted in low reached goal rates and validity rates. Although `ROBUST` again took the longest amount of computation time, it nevertheless produced solutions whose rollouts were nearly all successful. These results indicate that the proposed planning framework can incorporate a learned transition model successfully, while also satisfying the imposed robustness constraints.

6.2 Underactuated Adaptive Hand Experiments

In this section, we evaluate the algorithm for in-hand manipulation of underactuated adaptive hands. First, we study planning for a physics engine simulation of the hand following a real-hand demonstration.

We consider a two-finger adaptive hand fabricated through 3D printing [25]. The fingers are opposed to each other such that the hand achieves planar manipulation. Each finger has two compliant joints with springs. In addition, two actuators provide flexion to the fingers through tendons running along the length of each finger. The fingers also have high friction pads to avoid slipping. In [36], the position of the manipulated object and the actuators load were shown to be sufficient to describe the state of a real hand under a quasi-static motion assumption. Thus, in the following in-hand manipulation test cases, the state of the system is composed of the position and loads. In addition and for simplification, the experiments consider eight possible actions. An action is, in practice, unit changes to the angles of the actuators at each time step. That is, an action moves the two actuators with an angle vector of $\lambda(\gamma_1, \gamma_2)$ where λ is a predefined unit angle and γ_i is equal to either 1, -1 or 0 (yielding eight possible actions while excluding $(0,0)$).

6.2.1 Physics-Engine Experiments

We first evaluate the proposed planning algorithm on physics engine simulation of the Model T-42 adaptive hand [25] in Gazebo environment as seen in Figure 1. The compliance of the hand was modeled as proposed in [34]. Then, approximately $2,400,000$ transition points were recorded in 2.5 Hz while manipulating a cylinder with 19.2 mm mm diameter. Using the collected data, we have trained a Rectified Linear Unit (ReLU) BNN with two hidden layers and 200 neurons each. We also approximated the initial distribution $\mathbf{b}_o$ when grasping the object with a Gaussian.

We defined six goals within the workspace of the hand and positioned obstacles in the region. For each goal, a path was planned using `ROBUST`, `MEAN-ONLY` and `STANDARD` from $\mathbf{b}_o$ to the goal region. For each planned path, 10 rollouts of the action sequence were executed. The success rate and tracking errors are presented in Table 3. The scenarios, planned paths and rollouts are illustrated in Figure 5.

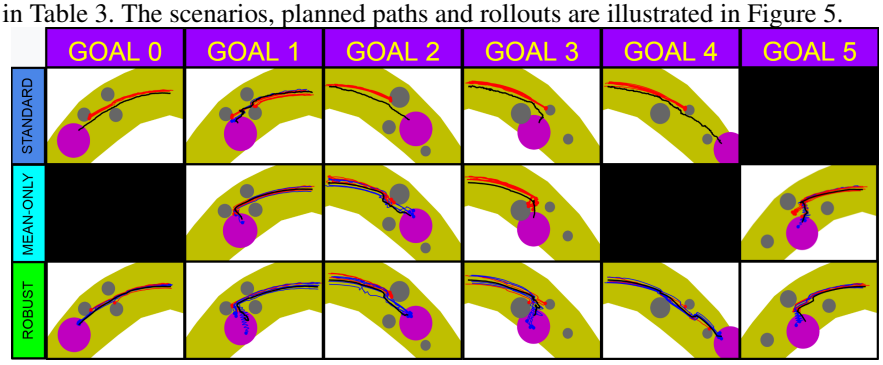

Fig. 5 Physics engine simulation results for manipulating the underactuated hand between obstacles (in gray). The black curves are the planned paths to the goal region (magenta circle). The blue and red curves are successful and failed paths, respectively. The yellow region is the approximated workspace of the hand. Black cells indicate that a solution was not found.

Table 3 Results for the Gazebo Adaptive Hand experiments averaged over 6 goals run twice each (12 trials).

Gazebo Adaptive Hand Experiments

	`STANDARD`	`MEAN-ONLY`	`ROBUST`
Initial Solution Time [s]	15.6	550.6	615.9
Initial Solution Path Length [mm]	51.28	55.98	67.01
Final Solution Path Length [mm]	49.73	50.12	65.42
Planning Solved Rate	83.3	58.3	91.7
Reached Goal Success Rate	0.04	0.25	.62
Validity Rate	0.06	.78	.73

Remarks: As shown in Table 3, `STANDARD` expectedly has the smallest computation time. Without any robustness constraints, `STANDARD` also produced the shortest solution paths. However, these paths typically approached the obstacles very closely. Consequently, as was shown in the prior experiments as well, the reached goal and validity rates of the `STANDARD` solutions were very low (4% and 6% respectively). Although `MEAN-ONLY` solutions improved these metrics (25% reached the goal and 78% were valid), this approach had the lowest planning solved rate with a much higher average computation time. The `ROBUST` method took the longest amount of

time, and also produced the longest solution paths. However, as also seen in the prior experiments, the ROBUST solutions had the highest rates of reaching the goal and remaining valid. Interestingly, ROBUST also had the highest planning solved rate, which seems to indicate that the robust constraints assist the exploration of the state space to reach the goal quicker and more efficiently.

6.2.2 Real Hand Demonstration

For a real hand demonstration, we have built an autonomous data collection system (seen in the supplementary video) to collect a sufficient amount of data. A three-finger Model-O underactuated hand [25], modified to use only two opposing fingers, was mounted on an arm of a Motoman DA20 Dual arm robot. It manipulated a cylinder with 45 mm diameter. To estimate the position of the cylinder during manipulation, ArUco fiducial markers [11] were attached to the hand and object such that, during in-hand manipulation, the position of the object relative to the hand is recorded. At each episode, the object is grasped, and manipulated with random actions while recording the state until dropped. After each drop of the object, a wire running through the object is stretched using the second arm of the Motoman to reposition it between the fingers toward regrasp. Approximately 400,000 transition points were recorded in 10 Hz and used to train a GP local regressor of 100 nearest neighbors. A peg-in-a-hole problem is demonstrated where the hand is tasked to drop the cylinder into a 45 mm diameter hole. Planned paths from the STANDARD and ROBUST algorithms were rolled-out and snapshots are seen in Figure 6.

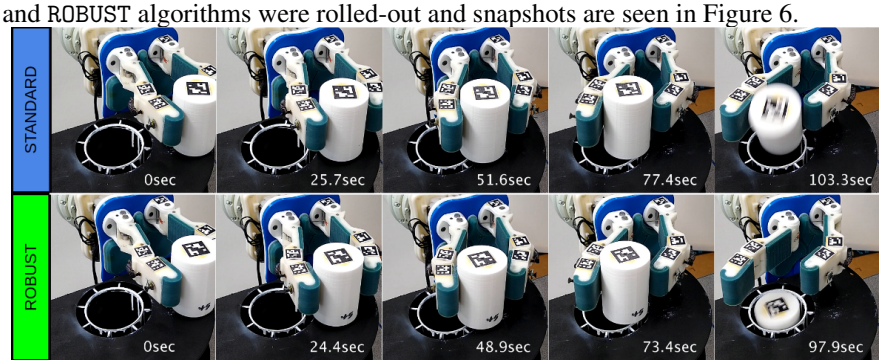

Fig. 6 Demonstration of STANDARD and ROBUST plans on the three-finger Model-O underactuated hand [25] for a peg-in-the-hole task.

7 Discussion

This paper proposes a belief-space planning framework, which utilizes data-based models. We combined the notion of particle propagation to consider distribution of states along with stochastic models that reason about uncertainty in the data. This combination was embedded in a sampling-based motion planning algorithm. We imposed constraints on the particles along the path such that the minimum predicted probability of success is above a predefined threshold. We also included a pruning condition, a cost-to-go heuristic and prioritized actions with better heuristics, all to improve the overall performance. A varying set of experiments, including an application for in-hand manipulation with underactuated hands, demonstrate that

the proposed algorithm showcase significant improvement in success rate relative to alternatives that do not reason about the underlying uncertainty. The key limitation is the higher computation time due to the need to propagate a large set of particles. In future work we will put effort in reducing the computation time for a batch of particles by utiling both systems-based approaches (e.g., parallelization) as well as algorithmic tools. We will also consider the expansion of the algorithm to path tracking, which can enable tasks such as writing characters using within-hand manipulation.

References

[1] de Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Annals of Operations Research **134** (2004)

[2] Bry, A., Roy, N.: Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty. In: ICRA (2011)

[3] Calli, B., Kimmel, A., Hang, K., Bekris, K., Dollar, A.: Path planning for within-hand manipulation over learned representations of safe states. In: International Symposium on Experimental Robotics. Buenos Aires, Argentina (2018)

[4] Calli, B., Srinivasan, K., Morgan, A., Dollar, A.M.: Learning modes of within-hand manipulation. In: ICRA, pp. 3145–3151 (2018). DOI 10.1109/ICRA.2018.8461187

[5] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion. The MIT Press (2005)

[6] Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS'18, pp. 4759–4770. Curran Associates Inc., USA (2018)

[7] Datta, A., Banerjee, S., Finley, A.O., Gelfand, A.E.: Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. J. of the American Stat. Ass. **111**(514), 800–812 (2016)

[8] Deisenroth, M.P., Rasmussen, C.E.: Pilco: A model-based and data-efficient approach to policy search. In: ICML (2011)

[9] Finn, C., Levine, S.: Deep visual foresight for planning robot motion. In: ICRA, pp. 2786–2793. IEEE (2017)

[10] Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: ICML (2016)

[11] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition **47**(6), 2280–2292 (2014)

[12] Girard, A.: Approximate methods for propagation of uncertainty with gaussian process models. Ph.D. thesis, University of Glasgow (2004)

[13] van Hoof, H., Hermans, T., Neumann, G., Peters, J.: Learning robot in-hand manipulation with tactile features. In: HUMANOIDS (2015)

[14] Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R.H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., Michalewski, H.: Model-based reinforcement learning for atari (2019)

[15] Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. IJRR **30**(7), 846–894 (2011)

[16] Kneebone, M., Dearden, R.: Navigation Planning in Probabilistic Roadmaps with Uncertainty. In: Proc. International Conference on Automated Planning and Scheduling (2009)

[17] Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In: RSS (2008)

[18] Levine, S., Koltun, V.: Guided policy search. In: ICML, pp. III–1–III–9 (2013)

[19] Li, Y., Littlefield, Z., Bekris, K.E.: Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning. In: WAFR (2014)
[20] Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. International Journal of Robotics Research (IJRR) **35**(5), 528–564 (2016)
[21] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: ICLR (2016)
[22] Littlefield, Z., Bekris, K.E.: Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In: IROS. Madrid, Spain (2018)
[23] Littlefield, Z., Klimenko, D., Kurniawati, H., Bekris, K.E.: The importance of a suitable distance function in belief-space planning. In: A. Bicchi, W. Burgard (eds.) International Symposium on Robotic Research (ISRR), pp. 683–700 (2015)
[24] Ma, R.R., Bircher, W.G., Dollar, A.M.: Toward robust, whole-hand caging manipulation with underactuated hands. In: IEEE Int. Conf. on Rob. and Aut., pp. 1336–1342 (2017)
[25] Ma, R.R., Dollar, A.M.: Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption. IEEE Rob. & Aut. Mag. **24**, 32–40 (2017)
[26] MacKay, D.J.C.: A practical bayesian framework for backpropagation networks. Neural Computation **4**(3), 448–472 (1992)
[27] McAllister, R., Rasmussen, C.E.: Improving pilco with bayesian neural network dynamics models (2016)
[28] Mchutchon, A., College, C.: Nonlinear modelling and control using gaussian processes (2014)
[29] Melchior, N.A., Simmons, R.: Particle RRT for Path Planning with Uncertainty. In: ICRA, pp. 1617–1624 (2007)
[30] Nguyen-Tuong, D., Peters, J.: Local gaussian process regression for real-time model-based robot control. In: IROS, pp. 380–385 (2008)
[31] Piergiovanni, A.J., Wu, A., Ryoo, M.S.: Learning real-world robot policies by dreaming. CoRR (2018)
[32] Prentice, S., Roy, N.: The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. IJRR **28**(11-12), 1448–1465 (2009)
[33] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. Cambridge, MA, The MIT Press (2005)
[34] Rocchi, A., Ames, B., Li, Z., Hauser, K.: Stable simulation of underactuated compliant hands. In: ICRA, pp. 4938–4944 (2016)
[35] Rybkin, O., Pertsch, K., Jaegle, A., Derpanis, K.G., Daniilidis, K.: Unsupervised learning of sensorimotor affordances by stochastic future prediction. CoRR (2018)
[36] Sintov, A., Morgan, A.S., Kimmel, A., Dollar, A.M., Bekris, K.E., Boularias, A.: Learning a state transition model of an underactuated adaptive hand. IEEE Robotics and Automation Letters **4**(2), 1287–1294 (2019)
[37] Spong, M.W.: Underactuated mechanical systems. In: B. Siciliano, K.P. Valavanis (eds.) Control Problems in Robotics and Automation, Lecture Notes in Control and Information Sciences (1997)
[38] Thrun, S.: Monte-Carlo POMDPs. In: NIPS (2000)
[39] Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M.: Embed to control: A locally linear latent dynamics model for control from raw images (2015)