# Guided Policy Search for In-Hand Manipulation

Bachelor Thesis of Tim Schneider
Date of submission: October 9, 2018

Examiner: Prof. Jan Peters
Supervisors: Filipe Veiga, Hany Abdulsamad

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Department of Computer Science
Intelligent Autonomous Systems

Guided Policy Search for In-Hand Manipulation

Bachelor Thesis submitted by Tim Schneider

Examiner: Prof. Jan Peters
Supervisors: Filipe Veiga, Hany Abdulsamad

Date of submission: October 9, 2018

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Tim Schneider, die vorliegende Bachelor Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Datum / Date: Unterschrift / Signature:

_____ _____

## Abstract

Dexterous in-hand manipulation allows robots to precisely reconfigure objects at a low time and energy effort. However, the complex contact-rich dynamics of in-hand manipulation are hard to model and uncertainty must be taken into account. We describe a reinforcement learning method that does not require an analytical model but estimates the dynamics using samples of policy roll-outs. To deal with uncertainty, we utilize high-dimensional tactile sensors on the fingertips.

Since tactile sensor data is hard to interpret, we perform supervised end-to-end training of a neural network policy, mapping the sensor inputs directly to torque commands. To generate training data for the network, a trajectory-centric reinforcement learning algorithm is deployed. This algorithm produces local controllers which are able to solve the problem for specific initial states. Trajectories resulting from these controllers are then used to train the neural network in a supervised learning fashion. Since these trajectories succeed from different initial states, the neural network is able to generalize and solve the problem for unseen states.

The method is evaluated in simulation on three different tasks: positioning fingers, rolling a cylinder, and positioning a freely-moving object. We present results for all tasks and conclude that our method is capable of learning dexterous manipulation skills.

## Contents

## 1 Introduction

The evolution of the human hand and our cognitive ability to control it is believed to be one of the major reasons for our success as a species [1]. Our hands play a crucial role in nearly all our physical interaction with the environment and enable us to perform a variety of tasks, including grasping, movement of objects, tool usage, or even complex in-hand manipulation like twirling a pen. Regarding robotics, in-hand manipulation can be vital to reorient objects that cannot be grasped in their desired configuration or need to be reconfigured during the execution of a task. Especially, when the movement range of the robot's arm is limited due to obstacles and some orientations cannot be reached, it can be beneficial to perform object reorientation in-hand. Additionally, hand dexterity can improve precision and efficiency, since a lower inertial load needs to be moved when using the fingers, compared to using the entire arm [2]. As an example, consider a robot using a screwdriver to remove a screw. While dexterity is arguably already required to insert the screwdriver into the screw, it also becomes important when the robot is twisting the screw. Due to joint limits and obstacles, the robot might be incapable of performing a full screwdriver rotation using its arm only. To avoid having to remove the screwdriver from the screw to reorient it, dexterity could be used to reconfigure it at a lower time and energy effort. However, despite numerous possibilities for robotic applications, dexterous in-hand manipulation is a largely unsolved problem in robotics.

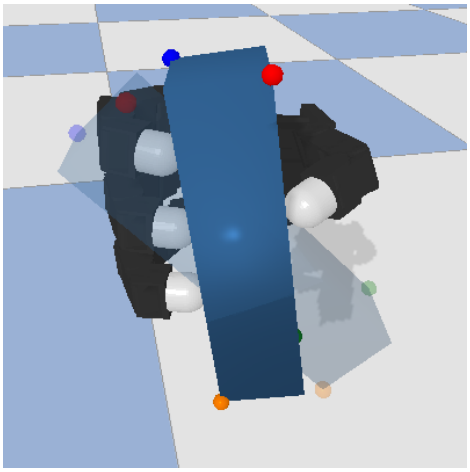

**Figure 1:** Allegro hand manipulating a freely-moving cuboid in simulation.

In this work, we describe our approach on tackling the problem of in-hand manipulation in a general way. A major challenge of hand-based manipulation tasks are the complex discontinuous dynamics that arise from contact-rich environments. Due to their complexity, these dynamics are hard to model accurately, making trajectory planning for hand-based manipulation often infeasible. Jointly with the underactuation that naturally occurs when manipulating objects, the complex dynamics make controller design for in-hand manipulation a difficult problem. Thus, approaches which use planning typically focus on very specific manipulation tasks and strategies on low dimensional systems for which analytical models can be derived [3–12]. Since we aim to tackle the problem of in-hand manipulation in a general way, our method does not require an analytical model of the system. Instead, we rely on a reinforcement learning algorithm that estimates a dynamics model using samples taken from the robot. The objective is defined in terms of a high level cost function that merely specifies the desired goal but does not contain any robot or solution specific knowledge. Consequently, our algorithm is not limited to predefined strategies or tasks but tackles in-hand manipulation from a general perspective.

However, additionally to the complex dynamics, the high number of continuous control and state dimensions make reinforcement learning challenging. Since discretizing the action space is not a good option for high numbers of dimensions, our method relies on a policy gradient. In prior work, policy gradient methods have been shown to be capable of dealing with different manipulation tasks [13–15], including in-hand manipulation [16–19]. The ability to deal with high-dimensional continuous systems makes policy gradient methods a reasonable choice for in-hand manipulation.

Another key challenge of hand-based manipulation is uncertainty. In systems with highly nonlinear dynamics, small perturbations in the state can already lead to entirely different outcomes. As an example, consider the screwdriver robot holding the screwdriver slightly displaced and thus missing the screw with its tip. While without the displacement, the robot would succeed in removing the screw, the robot now might not even manage to perform a single rotation of the screw. One way to deal with uncertainty is the use of compliant hands, which mechanically adapt to the manipulated object's geometry [20, 21]. However, Jentoft et al. [21] point out that there are limits to compliance, particularly when the uncertainty in object size or weight is high. To overcome these limits, they suggest to use tactile sensors and show improvements in grasping precision and stability. Prior work has also shown that tactile sensors are not only capable of improving grasping [22–29], but also provide enough data to perform manipulation [16, 18, 27, 30–33]. In our work, our final control policy relies solely on tactile feedback, which we found sufficient to perform manipulation tasks.

While it is possible to train a policy directly on sensor data, like tactile data [16], training is likely to become significantly faster when information about the full system state is included [34]. For in-hand manipulation, these information can include, for example, the pose of the manipulated object. While this information can be acquired during training time, for example by using motion trackers, it is unlikely that it will be accessible outside of the training environment. In other words, requiring objects to have motion tracking tags to be able to manipulate them is imposing a serious limitation to the

method's usability. To overcome these limitations while still taking advantage of all information available at training time, we use a dual training approach. Using the full system state as input, we deploy a trajectory-centric policy gradient algorithm to generate good trajectories resulting from different initial states. These trajectories are then used to train a neural network policy in supervised learning fashion. Unlike the algorithm that generated the trajectories, the neural network receives only partial state information, like the joint states and tactile sensor data. This enables the resulting policy to be deployed outside controlled environments, where the full state is not available. Since the policy is trained using trajectories resulting from different initial states, it is able to generalize and reach the objective starting from previously unseen initial states. This method was proposed in prior work [34] and has been applied to different manipulation tasks [13, 19].

In order to evaluate this method, we designed three tasks with different objectives: positioning of the fingers, rolling a fixed cylinder, and positioning a freely-moving object. For each of these tasks, we present results of local controllers which solve the problem for specific initial states, and a global policy that generalizes and solves the problem for arbitrary initial states, including previously unseen states. All our experiments run in simulation, which enables us to perform rapid testing while providing a reasonable insight into the methods performance. We use a simulated version of the Allegro hand [35] with tactile sensors on the fingertips.

In the following section, we present related work on robotic manipulation and specifically in-hand manipulation. Section 3 contains a complete derivation of the algorithm used in this work. In Section 4, we provide details on the simulation environment we use and describe the setup of the experiments performed to validate our method. The results of the experiments are collectively summarized in the subsequent section. Finally, we draw a conclusion of our work and discuss possible future work in Section 6.

## 2 Related Work

**Planning-based approaches**

Many approaches to in-hand manipulation rely on known robot models to plan strategies in advance. Planning based approaches have been successfully applied to reposition objects in-hand by rolling them between fingers [4, 7, 8, 32], using finger gaiting [5], utilizing controlled rolling [3] or slip [11] induced by gravity or by pushing them against static objects in the environment [10, 12]. While these strategies are all very specific and limited to certain applications, there have also been efforts to use planning for more general tasks.

Hertkorn et al. [9] propose a method to plan grasping and subsequent reconfiguration of arbitrary objects. However, this method requires not only precise dynamics models of the robot and the object, but also a detailed definition of the desired object trajectory. Since not all object trajectories are feasible, these trajectories need to be carefully chosen for each object and initial condition, which limits the method's generality. Additionally, to find the ideal grasp, the method relies on iterating over all possible grasps, which increase rapidly with the number of fingers and dimensions. Consequently, Hertkorn et al. were able to present results in 2D environments only. In contrary, our method does not require any solution-specific input. Instead, we rely on a high-level description of the target state only, which enables the algorithm to generalize and solve the problem for previously unseen states. Furthermore, our method deals more efficiently with high-dimensionality and we show results using 4 fingers in 3 dimensions.

Similar to our method, other planning-based methods rely on high-level descriptions of the objective as well. Erez et al. [36] apply an iterative linear quadratic Gaussian regulator [37] to object manipulation and present results in manipulating a cylindric object using a 4 fingered non-anthropomorphic robot. Further, Mordatch et al. [38] use Contact Invariant Optimization [39] to learn a variety of hand-based manipulation skills, like grasping, twirling a cylindric object and drawing, in simulation. However, especially when facing previously unseen situations, an accurate model of both the robot and the environment might not be present. Our method uses samples of previous rollouts to estimate the system dynamics and is thus able to learn manipulation tasks without the need for a prior dynamics model.

**Usage of tactile sensors**

Several approaches show that tactile sensors are capable of measuring [22–24] and enhancing [21, 25–29] grasp stability. Compared to purely vision based grasping approaches [40, 41], tactile sensors provide more reliable and accurate information on object geometry and contacts and do not suffer from occlusion [24]. Furthermore, prior work has shown that robots are capable of performing manipulation tasks, while solely relying on tactile feedback and internal sensors. These tasks range from 2D object translation [16, 30, 31] and object reorientation [18, 32] to gentle scraping of a surface [33] and screwing a light bulb [27]. Since tactile data is only available when the robot is in contact with the object, some approaches combine tactile sensors with vision-based sensors [24] or rangefinders [28]. In our work, however, we assume that the object has been grasped prior to the manipulation and rely solely on tactile sensors as input for our policy.

**Imitation learning**

The concept of learning from human demonstration has been applied to robotics several times with remarkable results. Starting from human gesture imitation [42], imitation learning has also been applied to manipulation [14, 19, 43, 44] and even to high-dimensional tasks like grasping [19, 27, 29] and in-hand manipulation [45]. Multiple approaches to in-hand manipulation use human demonstrations to support reinforcement learning algorithms [15, 17–19].

In this work, however, we chose not to use imitation learning to initialize our algorithms for several reasons. Firstly, since we are working in simulation, the demonstrator could not gain any tactile feedback and performing manipulation tasks based on visual feedback only is challenging and might lead to poor results. Secondly, the only measure of task performance is a cost function, which would leave the demonstrator without clear notion of the actual objective. Finally, the structure of the Allegro hand differs in many ways from the human hand. Not only does it have one finger less, but it also allows to control the proximal and distal joints independently and to rotate all fingers around their longitudinal axis. Since humans typically cannot perform these movements, this would leave the demonstrator unable to fully exploit the robot hand's workspace. Thus, demonstrations of strategies that work very well for humans are likely to be sub-optimal for the robot and could result in a bad initialization.

**Reinforcement learning**

Contrary to planning-based approaches, reinforcement learning (RL) does not rely on manually designed models of the robot and the environment. This gives RL a significant advantage in terms of generality and makes it very suitable for tasks in unknown environments. In the past years RL has been successfully applied to many tasks in robotic manipulation, including grasping [14, 46, 47], opening doors [47], screwing a lid on a bottle [13], and stacking LEGO bricks [15].

Since discretizing the action space quickly becomes infeasible in high dimensions, these approaches typically rely either on policy gradient methods or PI2 [48] to solve a continuous RL problem.

Several approaches in the past years use RL to tackle the problem of in-hand manipulation. Van Hoof et al. [16] use Relative Entropy Policy Search (REPS) [49] to learn time-independent control policies for rolling cylindric objects between two fingers. Similar to our method, tactile feedback serves as the only input to the final control policy and the entire training is performed end-to-end, with no hand-designed feature extraction in-between. However, the gripper used in this work is considerably less complex in terms of state and control dimensions than the hand used in our approach.

Rajeswaran et al. [17] use a policy gradient method to learn in-hand manipulation based on expert demonstrations. To overcome the problems with imitation learning mentioned mentioned before, a specialized, well-shaped cost function is used to train another RL algorithm to generate demonstrations. This limits the method's generality, since this cost function needs careful manual adaptation for each different task. Additionally, the resulting policy needs the exact object position as input, while our policy is trained to work on tactile sensor input only.

Similar to our work is the work of Kumar et al. [50], who learn to rotate a cylinder with an anthropomorphic hand. The method used in this work is the same method we use to generate our local controllers. However, these local controllers do not generalize and only work for specific initial states. Kumar et al. overcome this problem in a later publication [18], by introducing a nearest-neighbor method and alternatively a neural network trained as global policy using data of the local controllers. The difference to our work is that this policy is trained offline, after the training of the local controllers is complete. In our work, however, the global policy is trained jointly with the local controllers, while the KL divergence between them is limited to ensure that the local controllers provide a good training set.

Gupta et al. [19] propose an approach to hand-based manipulation that uses the same policy search method as in our work and report results in turning a valve, pushing beads on an abacus and grasping. Contrary to our work, they aim to imitate a human demonstrator performing a task while we define our tasks exclusively in terms of the objective to be reached. Additionally, Gupta et al. need motion trackers to feed the object position into the policy at test time, while we only require tactile input for testing.

## 3 Guided Policy Search

This section describes the policy search algorithm we use to learn control policies for the Allegro hand. Guided Policy Search (GPS) was proposed by Levine et al. and published in multiple iterations over the past years [13, 34, 51–54]. This section aims to present the derivations of the latest iteration [34] in a comprehensive way, by adding details and intermediate steps omitted in the original work.

### 3.1 Problem Specification and Definitions

The objective of stochastic policy search is to find a policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ that minimizes the expected value of a cost function $\ell\left(\tau\right) = \sum_{t=1}^{T} \ell_t\left(\mathbf{u}_t, \mathbf{x}_t\right)$ in an episode of $T$ discrete time steps. Here, $\theta$ denotes the policy parameters to optimize, for example the weights of a neural network representing the policy. Furthermore, $\tau = \left(\mathbf{x}_1, \mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{x}_T, \mathbf{o}_T, \mathbf{u}_T\right)$ denotes a trajectory, which is modelled as a sequence of states $\mathbf{x}_t$, observations $\mathbf{o}_t$, and actions $\mathbf{u}_t$. More specifically, $\mathbf{x}_t$ is a real valued vector representation of the full system state at time $t$, which could contain robot joint angles and velocities, as well as positions of objects in the scene. $\mathbf{o}_t$ is a vector of observations that serves as feedback for the policy $\pi_\theta$. The observation vector can contain similar entities as the state vector but does not contain any information that is not available at test time. Thus, $\mathbf{o}_t$ could contain tactile sensor data but not the exact position of objects in the scene, since these can be difficult to acquire in unstructured environments. $\mathbf{u}_t$ is a vector representation of the performed action that led to the transition of state $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. As an example, $\mathbf{u}_t$ could contain torque commands sent to the motor at time step $t$. Note that the last action of a trajectory $\mathbf{u}_T$ has no influence on the state trajectory, since no state is following, and could be omitted. However, we chose to use this representation for notational convenience and will assume that $\mathbf{u}_T$ is constant 0. The observation distribution $P_t\left(\mathbf{o}\middle|\mathbf{x}\right)$ as well as the real system dynamics $P_t\left(\mathbf{x}'\middle|\mathbf{u}, \mathbf{x}\right)$ are assumed to be unknown.

While the observations $\mathbf{o}$ serve as feedback for the global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$, they do not have any direct influence on the cost function $\ell$. Hence, they can be omitted when measuring the policy's performance. In the following, we write $\hat{\tau}$ for state-action trajectories that do not contain observations, such that $\hat{\tau} = \left(\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_T, \mathbf{u}_T\right)$. Jointly with the system dynamics, the policy induces a distribution over state-action trajectories $\hat{\tau}$:

$$\pi_\theta\left(\hat{\tau}\right) = P_1\left(\mathbf{x}_1\right) \prod_{t=1}^{T-1} P_t\left(\mathbf{x}_{t+1}\middle|\mathbf{u}_t, \mathbf{x}_t\right) \pi_\theta\left(\mathbf{u}_t\middle|\mathbf{x}_t\right)$$

where $P_1\left(\mathbf{x}_1\right)$ is the distribution of the initial state $\mathbf{x}_1$ and $\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right) = \int \pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right) P_t\left(\mathbf{o}\middle|\mathbf{x}\right) d\mathbf{o}$ is the global policy conditioned on the state. Using this definition, the overall objective becomes:

$$\min_\theta \mathop{\mathbb{E}}_{\pi_\theta\left(\hat{\tau}\right)} \left[\ell\left(\hat{\tau}\right)\right] \tag{1}$$

### 3.2 Method Summary

GPS splits the problem of policy search into two components. The first component is a trajectory-centric RL algorithm that learns to solve the problem for multiple distinct initial states. Considering in-hand manipulation, these initial states could correspond to different poses of the manipulated object relative to the hand. In the following, we will use index $i \in \{1, \ldots, N\}$ as a superscript to refer to the $i$-th initial state. Consequently, we write $P_1^i\left(\mathbf{x}\right)$ for the $i$-th initial state distribution. For each initial state distribution $P_1^i\left(\mathbf{x}\right)$, a time-varying local controller $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ is trained to minimize the expected value of the cost function $\ell\left(\tau^i\right)$ for trajectories $\tau^i$ succeeding from this initial state distribution. It is important to note that each controller is trained on its specific initial state distribution only and is unlikely to succeed from states outside that distribution. Thus, these controllers alone are not sufficient to solve any task in general.

However, the local controllers can be used to generate training data for a supervised learning algorithm, which forms the second component of GPS. The objective of the second component is to train a single time-independent global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ to resemble the behavior of all local controllers. Trajectories $\tau_j^i$ generated by rolling out the local controllers on the robot multiple times serve as demonstrations. The subscript $j$ is used here to differentiate between trajectories generated by different roll-outs of the same local controller. Note that unlike the local controllers, wich are conditioned on the full system state $\mathbf{x}$, the global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ is conditioned on the observation $\mathbf{o}$. This enables us to roll out the global policy outside of controlled training environments, where the full state $\mathbf{x}$ cannot be recorded and the robot has to rely on sensor observations.

In the supervised learning scenario, the local controllers serve as experts for trajectories resulting from their respective initial state distribution, while the global policy is the learner. The most straightforward way of transferring this

knowledge would probably be to train the local controllers until they reach good performance and use the resulting trajectories as training data for the global policy. This approach however has a flaw, as during roll-outs of the learner's policy, small compounding control errors can lead the learner into states that are far away from the expert's trajectory distribution. Since the experts demonstrate no knowledge for these states, the learner cannot learn to succeed from them and will perform poorly. Thus, to be successful for the expert's initial states alone, the learner would have to copy their trajectories exactly and is at that point likely to overfit. To tackle this problem, Ross et al. [55] suggest to apply the expert policy to states from the learner's state distribution. This way, the learner receives feedback on how to behave for states from its own state distribution instead of states from the expert's distribution. Here, a similar approach is chosen, by training local controllers and global policy in an alternating fashion, while at the same time enforcing a KL divergence constraint between them. This forces the local trajectory optimizer to remain close to the global policy and hence generate training data within the learner's state distribution.

Hence, the local controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ are optimized to minimize the cost function while staying close to the global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$, and the global policy is optimized to match the local controllers. Since $\tau_j^i$ contains trajectories succeeding from different initial states, the global policy is able to generalize and can be applied to previously unseen initial states. The efficiency of this method arises from the fact that the full system state is assumed to be known during training time. Knowledge of the state enables us to train the local controllers based on full state information, which is more efficient than training the global policy based on observations directly. However, during test time when the global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ is executed on the robot, knowledge of the full state $\mathbf{x}$ is not required, as the global policy was trained based on observations $\mathbf{o}$ only.

Figure 2 presents a flowchart of the GPS algorithm. Each iteration of the outer loop starts by rolling out all current local controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ on the robot. In the first iteration, these local controllers are initialized to be a Gaussian distribution around 0 with small variance. Since the dynamics of the system are unknown, the collected trajectories $\tau_j^i$ are used to estimate a local dynamics model $d_t^i\left(\mathbf{x}'\middle|\mathbf{u},\mathbf{x}\right)$ for each initial state. The inner loop alternates between optimizing the local controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ using the estimated dynamics $d_t^i\left(\mathbf{x}'\middle|\mathbf{u},\mathbf{x}\right)$ and optimizing the global policy $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ to match $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$. Note that in the inner loop, no new samples are drawn from the robot. Since the trajectories $\tau_j^i$ were drawn using the controllers of the previous iteration, only their states are utilized for training of the global policy, while the corresponding actions are sampled directly from the updated local controllers. The entire process is repeated until convergence.

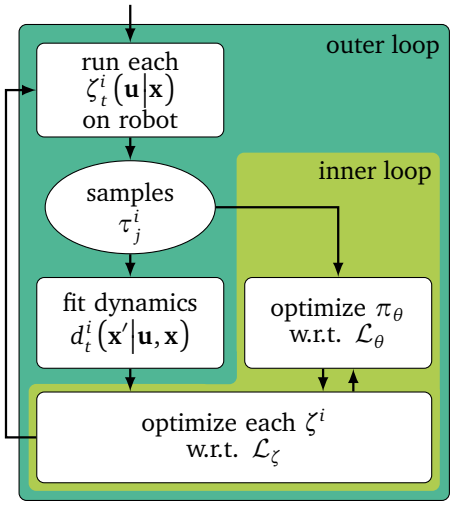


**Figure 2:** Flowchart of the GPS algorithm. $\mathcal{L}_\zeta$ and $\mathcal{L}_\theta$ are the Lagrangians of the local and global optimization as further defined in Section 3.3.

In the following, we will use the term "roll-out" exclusively to refer to a single execution of a local controller or the global policy on the robot. Each roll-out produces a trajectory $\tau$, which we will also refer to as "sample". The term "iteration" is reserved for an iteration of the outer loop as shown in Figure 2. Each iteration starts with drawing samples from the robot and results in an updated global policy and updated local controllers, which are used to draw samples for the next iteration. Furthermore, we will use the term "current samples" to refer to the samples drawn at the beginning of the current iteration and "previous samples" to refer to samples drawn in all prior iterations.

### 3.3 Algorithm Derivation

Guided policy search aims to minimize the expected cost of trajectories generated by the local controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$, while ensuring that the global policy matches all local controllers:

$$\min_{\zeta,\theta}\sum_{i=1}^{N}\mathop{\mathbb{E}}_{\zeta^i\left(\hat{\tau}\right)}\left[\ell\left(\hat{\tau}\right)\right]\quad\text{s.t.}\quad\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)=\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)\quad\forall i\in\mathbb{I},t\in\mathbb{T},\mathbf{u}\in\mathbb{R}^{n_u},\mathbf{x}\in\mathbb{R}^{n_x}\tag{2}$$

where $\mathbb{I}=\{1,\ldots,N\}$ is the set of initial state identifiers, $\mathbb{T}=\{1,\ldots,T\}$ is the set of time steps of a single controller roll-out and $n_u$ and $n_x$ are the dimensions of the action vector $\mathbf{u}$ and the state vector $\mathbf{x}$. Furthermore, the probability distribution of state-action trajectories under the local controller $\zeta^i\left(\hat{\tau}\right)$ is defined as

$$\zeta^i\left(\hat{\tau}\right)=P_1^i\left(\mathbf{x}_1\right)\prod_{t=1}^{T-1}d_t\left(\mathbf{x}_{t+1}\middle|\mathbf{u}_t,\mathbf{x}_t\right)\zeta_t^i\left(\mathbf{u}_t\middle|\mathbf{x}_t\right)$$

Recall that $P_1^i\left(\mathbf{x}_1\right)$ is the initial state distribution of controller $i$. For completeness, we define the state distribution of the local controllers at time $t+1$ in a recursive manner as

$$\begin{aligned}\zeta_{t+1}^i\left(\mathbf{x}_{t+1}\right)&:=\int\!\!\int_{-\infty}^{\infty}d_t\left(\mathbf{x}_{t+1}\middle|\mathbf{u}_t,\mathbf{x}_t\right)\zeta_t^i\left(\mathbf{x}_t,\mathbf{u}_t\right)d\mathbf{x}_t d\mathbf{u}_t,\quad\text{where}\\ \zeta_t^i\left(\mathbf{x}_t,\mathbf{u}_t\right)&:=\zeta_t^i\left(\mathbf{u}_t\middle|\mathbf{x}_t\right)\zeta_t^i\left(\mathbf{x}_t\right)\\ \zeta_1^i\left(\mathbf{x}_1\right)&:=P_1^i\left(\mathbf{x}_1\right)\end{aligned}$$

Since the constraint ensures equality between the local controller and global policy, minimizing w.r.t to (2) is equivalent to minimizing w.r.t the objective (1). Note that even though the constraint is defined on $\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)$, we optimize $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ and thus control $\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)$ indirectly, since $\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)=\int\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)P_t\left(\mathbf{o}\middle|\mathbf{x}\right)d\mathbf{o}$. If the observation $\mathbf{o}$ holds enough information about the state $\mathbf{x}$, controlling $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ is sufficient to satisfy the equality constraint. However, knowledge about $P_t\left(\mathbf{o}\middle|\mathbf{x}\right)$ is not required by this method.

In this implementation, the constrained dual problem (2) is solved using a variant of Bregman-ADMM (BADMM) [56]. Like ADMM, BADMM is a method for solving constrained dual problems of the form

$$\min_{\mathbf{v}\in\mathcal{V},\mathbf{w}\in\mathcal{W}}f\left(\mathbf{v}\right)+g\left(\mathbf{w}\right)\quad\text{s.t.}\quad\mathbf{A}\mathbf{v}+\mathbf{B}\mathbf{w}=\mathbf{c}$$

where $f$ and $g$ are convex functions, $\mathbf{A}\in\mathbb{R}^{m\times n_1}$, $\mathbf{B}\in\mathbb{R}^{m\times n_2}$, $\mathbf{c}\in\mathbb{R}^m$ and $\mathcal{X}\subseteq\mathbb{R}^{n_1}$ and $\mathcal{W}\subseteq\mathbb{R}^{n_2}$ are nonempty closed convex sets. However, BADMM allows to replace the squared Euclidean distance in ADMM's Lagrangian by a Bregman divergence $B$ to exploit the problem structure and offers better convergence guarantees in return. This yields the following dual update step:

$$\begin{aligned}\mathbf{v}^{(i+1)}&=\mathop{\arg\min}_{\mathbf{v}\in\mathcal{V}}f\left(\mathbf{v}\right)+\left\langle\lambda^{(i)},\mathbf{A}\mathbf{v}+\mathbf{B}\mathbf{w}^{(i)}-\mathbf{c}\right\rangle+\rho B\left(\mathbf{c}-\mathbf{A}\mathbf{v},\mathbf{B}\mathbf{w}^{(i)}\right)\\ \mathbf{w}^{(i+1)}&=\mathop{\arg\min}_{\mathbf{w}\in\mathcal{W}}g\left(\mathbf{w}\right)+\left\langle\lambda^{(i)},\mathbf{A}\mathbf{v}^{(i)}+\mathbf{B}\mathbf{w}-\mathbf{c}\right\rangle+\rho B\left(\mathbf{B}\mathbf{w},\mathbf{c}-\mathbf{A}\mathbf{v}^{(i+1)}\right)\\ \lambda^{(i+1)}&=\lambda^{(i)}+\alpha\left(\mathbf{A}\mathbf{v}^{(i)}+\mathbf{B}\mathbf{w}^{(i)}-\mathbf{c}\right)\end{aligned}$$

where $\lambda\in\mathbb{R}^m$ is the Lagrangian multiplier, $\rho\in\mathbb{R}$ is a penalty factor and $\alpha\in\mathbb{R}$ is the update rate of the Lagrangian multiplier.

To transfer the original optimization problem into a problem we can solve using BADMM, we start by modifying the equality constraint in (2):

$$\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)\zeta_t^i\left(\mathbf{x}\right)-\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)\zeta_t^i\left(\mathbf{x}\right)=0\quad\forall i\in\mathbb{I},t\in\mathbb{T},\mathbf{u}\in\mathbb{R}^{n_u},\mathbf{x}\in\mathbb{R}^{n_x}\tag{3}$$

While this constraint is still equivalent to the original constraint, it will simplify the update rules as we will see later. Since it is not possible to express this constraint as a linear function of finite dimensional vectors $\mathbf{A}\mathbf{v}+\mathbf{B}\mathbf{w}=\mathbf{c}$, we need to introduce modifications to the original BADMM formulation. We start by redefining $\mathcal{V}$ and $\mathcal{W}$ to fit our needs.

Since $\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})$ as well as $\zeta_t^i(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})$ are both time-varying probability density functions over state-action pairs conditioned on the respective initial state $i$, we define $\mathcal{V}$ and $\mathcal{W}$ accordingly:

$$\mathcal{V} := \mathcal{W} := \left\{ f : \mathbb{I} \times \mathbb{T} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R} \;\middle|\; \forall i \in \mathbb{I} : \forall t \in \mathbb{T} : \left( \forall \mathbf{u} \in \mathbb{R}^{n_u} : \forall \mathbf{x} \in \mathbb{R}^{n_x} : f_t^i(\mathbf{u},\mathbf{x}) \geq 0 \wedge \int_{-\infty}^{\infty} f_t^i(\mathbf{u},\mathbf{x})\, d\mathbf{u} = 1 \right) \right\} \tag{4}$$

This modification also implies that the matrices $\mathbf{A}$ and $\mathbf{B}$ need to be replaced by linear functions $a$ and $b$ that take functions in $\mathcal{V}$ or $\mathcal{W}$ as arguments and return arbitrary functions with consistent signatures. Furthermore, $\mathbf{c}$ must be replaced with a function $c$ consistent with the output functions of $a$ and $b$. Setting $a(\mathbf{v}) := \mathbf{v}$, $b(\mathbf{w}) := -\mathbf{w}$ and $c := 0$ and setting $\mathbf{v}_t^i(\mathbf{u},\mathbf{x}) = \pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})$ as well as $\mathbf{w}_t^i(\mathbf{u},\mathbf{x}) = \zeta_t^i(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})$ yields the modified equality constraint (3).

We define the optimization objective function of the local controller to equal the expected cost of the local controllers:

$$\begin{aligned} g(\mathbf{w}) &:= \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\ell_t(\mathbf{x},\mathbf{u})\right] \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\ell_t(\mathbf{x},\mathbf{u})\right] \\ &= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\zeta^i(\hat{\tau})}\left[\ell(\hat{\tau})\right] \end{aligned}$$

$f(\pi_\theta)$ remains zero as the cost function $\min_{p,\theta} \mathbb{E}_{\zeta(\tau)}[\ell(\tau)]$ does not depend on the global policy. Consequently, the resulting update steps of the dual descent algorithm are:

$$\begin{aligned} \mathbf{v} &\leftarrow \arg\min_{\mathbf{v}} \langle \lambda, \mathbf{v} - \mathbf{w} \rangle + \rho B(\mathbf{v},\mathbf{w}) \\ \mathbf{w} &\leftarrow \arg\min_{\mathbf{w}} \left( \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\ell_t(\mathbf{x},\mathbf{u})\right] \right) + \langle \lambda, \mathbf{v} - \mathbf{w} \rangle + \rho B(\mathbf{w},\mathbf{v}) \\ \lambda &\leftarrow \lambda + \alpha(\mathbf{v} - \mathbf{w}) \end{aligned} \tag{5}$$

Note that, due to (5) and the definition of $\mathbf{v}$ and $\mathbf{w}$, $\lambda$ is a function $\mathbb{I} \times \mathbb{T} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R}$.

Since all elements in $\mathcal{V}$ and $\mathcal{W}$ are integrable by definition, a scalar product can now be defined as:

$$\langle f, g \rangle := \sum_{i=1}^{N}\sum_{t=1}^{T} \int\int_{-\infty}^{\infty} f_t^i(\mathbf{u},\mathbf{x})\, g_t^i(\mathbf{u},\mathbf{x})\, d\mathbf{u} d\mathbf{x}$$

Using this definition and writing $\lambda_{i,t,\mathbf{x},\mathbf{u}}$ for $\lambda(i,t,\mathbf{x},\mathbf{u})$, the update steps become:

$$\mathbf{v} \leftarrow \arg\min_{\mathbf{v}} \sum_{i=1}^{N}\sum_{t=1}^{T} \int\int_{-\infty}^{\infty} \lambda_{i,t,\mathbf{x},\mathbf{u}} \left( \mathbf{v}_t^i(\mathbf{u},\mathbf{x}) - \mathbf{w}_t^i(\mathbf{u},\mathbf{x}) \right) d\mathbf{x} d\mathbf{u} + \rho B(\mathbf{v},\mathbf{w}) \tag{6}$$

$$\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\ell_t(\mathbf{x},\mathbf{u})\right] + \int\int_{-\infty}^{\infty} \lambda_{i,t,\mathbf{x},\mathbf{u}} \left( \mathbf{v}_t^i(\mathbf{u},\mathbf{x}) - \mathbf{w}_t^i(\mathbf{u},\mathbf{x}) \right) d\mathbf{x} d\mathbf{u} + \rho B(\mathbf{w},\mathbf{v}) \tag{7}$$

$$\lambda_{i,t,\mathbf{x},\mathbf{u}} \leftarrow \lambda_{i,t,\mathbf{x},\mathbf{u}} + \alpha\left( \mathbf{v}_t^i(\mathbf{u},\mathbf{x}) - \mathbf{w}_t^i(\mathbf{u},\mathbf{x}) \right) \quad \forall i \in \mathbb{I}, t \in \mathbb{T}, \mathbf{u} \in \mathbb{R}^{n_u}, \mathbf{x} \in \mathbb{R}^{n_x}$$

Due to the modification of the equality constraint (3), we can now express the integrals in (6) and (7) as expectations:

$$\mathbf{v} \leftarrow \arg\min_{\mathbf{v}} \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathbf{v}_t^i(\mathbf{u},\mathbf{x})}\left[\lambda_{i,t,\mathbf{x},\mathbf{u}}\right] - \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\lambda_{i,t,\mathbf{x},\mathbf{u}}\right] + \rho B(\mathbf{v},\mathbf{w}) \tag{8}$$

$$\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\ell_t(\mathbf{x},\mathbf{u})\right] + \mathop{\mathbb{E}}_{\mathbf{v}_t^i(\mathbf{u},\mathbf{x})}\left[\lambda_{i,t,\mathbf{x},\mathbf{u}}\right] - \mathop{\mathbb{E}}_{\mathbf{w}_t^i(\mathbf{u},\mathbf{x})}\left[\lambda_{i,t,\mathbf{x},\mathbf{u}}\right] + \rho B(\mathbf{w},\mathbf{v}) \tag{9}$$

As $\mathbf{v}$ and $\mathbf{w}$ are probability distributions, we select the Bregman divergence term $B$ to be the expectation of the KL divergence of the global policy and the local controller:

$$B(\mathbf{v},\mathbf{w}) := \sum_{i=1}^{N}\sum_{t=1}^{T} \phi_t^{\theta}\left(\theta,\zeta^i\right)$$
$$B(\mathbf{w},\mathbf{v}) := \sum_{i=1}^{N}\sum_{t=1}^{T} \phi_t^{\zeta}\left(\zeta^i,\theta\right)$$

where $\phi_t^{\theta}\left(\theta,\zeta^i\right)$ and $\phi_t^{\zeta}\left(\zeta^i,\theta\right)$ are the expected controller KL divergences at time $t$ under the local controller's trajectory distribution:

$$\phi_t^{\theta}\left(\theta,\zeta^i\right) := \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x})}\left[D_{KL}\left(\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right) \parallel \zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)\right)\right]$$
$$\phi_t^{\zeta}\left(\zeta^i,\theta\right) := \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x})}\left[D_{KL}\left(\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right) \parallel \pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)\right)\right]$$

We further replace the penalty multiplier $\rho$ by a time-varying multiplier $\upsilon_t$, which enables us to penalize divergence in certain parts of the trajectories more than in others. This is useful to enforce equality between local and global controller in specific regions where they start do diverge. Dropping terms independent of the optimization variable and replacing $\mathbf{v}$ and $\mathbf{w}$ with their definitions, this results in:

$$\theta \leftarrow \arg\min_{\theta} \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\lambda_{i,t,\mathbf{x},\mathbf{u}}\right] + \upsilon_t \phi_t^{\theta}(\theta,\zeta)$$
$$\zeta \leftarrow \arg\min_{\zeta} \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x},\mathbf{u})}\left[\ell_t\left(\mathbf{x},\mathbf{u}\right) - \lambda_{i,t,\mathbf{x},\mathbf{u}}\right] + \upsilon_t \phi_t^{\zeta}(\zeta,\theta)$$
$$\lambda_{i,t,\mathbf{x},\mathbf{u}} \leftarrow \lambda_{i,t,\mathbf{x},\mathbf{u}} + \alpha\upsilon_t\left(\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)\zeta_t^i\left(\mathbf{x}\right) - \zeta_t^i\left(\mathbf{x},\mathbf{u}\right)\right) \quad \forall i \in \mathbb{I}, t \in \mathbb{T}, \mathbf{u} \in \mathbb{R}^{n_u}, \mathbf{x} \in \mathbb{R}^{n_x}$$

As no assumptions about the class of the probability distributions have been made yet, $\zeta^i\left(\hat{\tau}\right)$ and $\pi_\theta\left(\hat{\tau}\right)$ can be chosen to be in a class of probability distributions that can be optimized efficiently. Choosing $\zeta^i\left(\hat{\tau}\right)$ to be Gaussian distributed enables us to use an Iterative Linear Quadratic Gaussian regulator (ILQG) [37] which optimizes trajectories in an highly efficient way, as described in Section 3.4. However, this requires not only the local controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ to be linear-Gaussian, but also implies assumes that the estimated system dynamics $d_t^i\left(\mathbf{x}'\middle|\mathbf{u},\mathbf{x}\right)$ are linear-Gaussian. While this seems like a drastic simplification, it is important to recall that the estimated dynamics are time-varying and we can select a different linear function for each time step. Time-varying linear functions enable to model arbitrary dynamics along a nominal trajectory and are sufficiently tolerant for real physical systems with small or Gaussian noise [34]. Even though the dynamics of contact-rich environments are usually highly non-linear, this approximation was found to to be sufficiently precise to learn in-hand manipulation skills [18, 50]. Due to the equality constraint between local and global controller in (2), $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ must be conditionally Gaussian like $\zeta_t\left(\mathbf{u}\middle|\mathbf{x}\right)$. Requiring the global policy to be conditionally Gaussian is reasonable, as mean and covariance can be any function of the observation $\mathbf{o}$, which allows $\pi_\theta\left(\mathbf{u}\middle|\mathbf{o}\right)$ to represent arbitrary policies with Gaussian noise.

Although these assumptions already introduce some simplifications to the original problem, a way to represent the infinite set of constraints $\pi_\theta\left(\mathbf{u}\middle|\mathbf{x}\right)\zeta_t^i\left(\mathbf{x}\right) = \zeta_t^i\left(\mathbf{x},\mathbf{u}\right)$ is still required. Peters et al. [49] proposed to approximate constraints on probability distributions by constraints on the expectation of features. If these features consist of monomials of the random variable, they introduce constraints on the moments of the distributions. In GPS, only the first order moments are used since that was found to be more stable than including higher moments [34]. Using the first order moments results in an equality constraint on the expected action: $\mathbb{E}_{\zeta_t^i(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\mathbf{u}\right] = \mathbb{E}_{\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\mathbf{u}\right]$. Although this approximation does not consider the action covariance of the controllers at all, it is still reasonable, as the previously assumed low entropy of the dynamics causes the covariance of the optimal control solution $\zeta(\tau)$ to be low. Additionally,

the KL-divergence terms $\phi_t^\theta\left(\theta,\zeta^i\right)$ and $\phi_t^\zeta\left(\zeta^i,\theta\right)$ will softly enforce equality between higher moments. The alternating optimization including the approximate constraints is given by:

$$\theta \leftarrow \arg\min_\theta \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\mathbf{u}^T\boldsymbol{\lambda}_{i,t}\right] + \upsilon_t\phi_t^\theta\left(\theta,\zeta\right) \tag{10}$$

$$\zeta \leftarrow \arg\min_\zeta \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x},\mathbf{u})}\left[\ell_t\left(\mathbf{x},\mathbf{u}\right) - \mathbf{u}^T\boldsymbol{\lambda}_{i,t}\right] + \upsilon_t\phi_t^\zeta\left(\zeta,\theta\right) \tag{11}$$

$$\boldsymbol{\lambda}_{i,t} \leftarrow \boldsymbol{\lambda}_{i,t} + \alpha\upsilon_t\left(\mathop{\mathbb{E}}_{\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t^i(\mathbf{x})}\left[\mathbf{u}\right] - \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x},\mathbf{u})}\left[\mathbf{u}\right]\right) \quad \forall i \in \mathbb{I}, t \in \mathbb{T}$$

where $\boldsymbol{\lambda}_{i,t}$ is the Lagrange multiplier for the expected action.

### 3.4 Local Controller Optimization under Unknown Dynamics

Local controller optimization is performed by a trajectory-centric RL algorithm, which is described in this section. The RL algorithm was proposed by Levine et al. [34] and can be seen as a variant of Iterative Linear Quadratic Gaussian regulator (ILQG). Objective of local controller optimization is to find time-varying controllers $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ that minimize the following Lagrangian for each starting condition $i \in \mathbb{I}$:

$$\mathcal{L}_\zeta^i\left(\zeta^i,\pi_\theta\right) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t^i(\mathbf{x},\mathbf{u})}\left[\ell_t\left(\mathbf{x},\mathbf{u}\right) - \mathbf{u}^T\boldsymbol{\lambda}_t\right] + \upsilon_t\phi_t^\zeta\left(\zeta^i,\theta\right) \tag{12}$$

Since the optimization procedure for each local controller is equal and independent of the other local controllers, the superscript of $\zeta_t^i\left(\mathbf{u}\middle|\mathbf{x}\right)$ is dropped for notational convenience and $\zeta_t\left(\mathbf{u}\middle|\mathbf{x}\right)$ is used instead.

As stated before, $\zeta\left(\tau\right)$ is chosen to be Gaussian distributed, which implies that the local controller $\zeta_t\left(\mathbf{u}\middle|\mathbf{x}\right)$ and the estimated dynamics $d_t\left(\mathbf{x}'\middle|\mathbf{u},\mathbf{x}\right)$ are both time-varying linear-Gaussian:

$$\zeta_t\left(\mathbf{u}\middle|\mathbf{x}\right) = \mathcal{N}\left(\mathbf{K}_t\mathbf{x} + \mathbf{k}_t|\boldsymbol{\Sigma}_t\right)$$
$$d_t\left(\mathbf{x}'\middle|\mathbf{u},\mathbf{x}\right) = \mathcal{N}\left(\mathbf{f}_{\mathbf{x},t}\mathbf{x} + \mathbf{f}_{\mathbf{u},t}\mathbf{u} + \mathbf{f}_{c,t}|\mathbf{F}_t\right)$$

where $\mathbf{K}_t \in \mathbb{R}^{n_u\times n_x}$, $\mathbf{k}_t \in \mathbb{R}^{n_u}$, $\boldsymbol{\Sigma}_t \in \mathbb{R}^{n_u\times n_u}$, $\mathbf{f}_{\mathbf{x},t} \in \mathbb{R}^{n_x\times n_x}$, $\mathbf{f}_{\mathbf{u},t} \in \mathbb{R}^{n_x\times n_u}$, $\mathbf{f}_{c,t} \in \mathbb{R}^{n_x}$, $\mathbf{F}_t \in \mathbb{R}^{n_x\times n_x}$ are the parameters of these distributions.

In this work, the dynamics are assumed to be unknown and need to be estimated using previously drawn samples. Using basic linear regression to obtain $\mathbf{f}_{\mathbf{x},t}$, $\mathbf{f}_{\mathbf{u},t}$, $\mathbf{f}_{c,t}$ and $\mathbf{F}_t$ from the latest samples works for systems with low state and action dimensions, but requires an infeasible amount of samples for higher dimensional systems, like the robot hand used in this work. To tackle this problem, a Normal-Inverse-Wishart distribution is used as a prior for the estimated system dynamics. The prior is obtained from a mixture of Gaussians, which is trained to fit the distribution of tuples $(\mathbf{x}_{t+1},\mathbf{u}_t,\mathbf{x}_t)$ from samples drawn in the current and previous iterations. Using a prior reduces the number of required samples drastically while still giving good approximations to the real system dynamics. In our work, 30 samples per iteration where found to be sufficient to approximate the dynamics of a 16 DOF robot hand controlling a freely-moving object. For further details to the dynamics fitting, please refer to Levine et al. [34].

The estimated dynamics are only valid in a small area around the sample trajectories they were estimated from, since they are only local approximations. Hence, the distance of the trajectories generated by the optimized controller to the sample trajectories needs to be bound, to ensure that the error of the estimated dynamics to the real dynamics remains low. Thus, similar to [49], an upper bound $\epsilon$ on the KL divergence of the new trajectory distribution to the distribution of the sample trajectories is introduced as a constraint and the objective becomes:

$$\min_\zeta \mathcal{L}_\zeta\left(\zeta,\pi_\theta\right) \quad \text{s.t.} \quad D_{KL}\left(\zeta^i\left(\hat{\tau}\right) \,\|\, \hat{\zeta}\left(\tau\right)\right) \leq \epsilon$$

where $\hat{\zeta}\left(\tau\right)$ is the distribution of the trajectories drawn during the sampling phase of the current iteration, thus using the local controller of the previous iteration. The upper bound $\epsilon$ of the KL divergence can be seen as a step size of the local controller optimization and needs to be chosen carefully.

As this objective introduces a new constraint, the Lagrangian is augmented with an additional term:

$$\mathcal{L}(\zeta) = \sum_{t=1}^{T} \left( \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x},\mathbf{u})} \left[ \ell_t(\mathbf{x},\mathbf{u}) - \mathbf{u}^T \boldsymbol{\lambda}_t \right] + \upsilon_t \phi_t^{\zeta}(\zeta,\theta) \right) + \eta \left( D_{KL}\left( \zeta(\tau) \parallel \hat{\zeta}(\tau) \right) - \epsilon \right) \tag{13}$$

where $\eta$ is the new Lagrangian multiplier. The overall algorithm now becomes an instance of generalized BADMM [56], as an additional Bregman divergence term is introduced by the new KL divergence constraint.

Note that the augmented Lagrangian $\mathcal{L}(\zeta)$ still depends on the global policy conditioned on the state $\pi_\theta(\mathbf{u}|\mathbf{x})$ due to the KL constraint $\phi_t^{\zeta}(\zeta,\theta)$ in (12). Unfortunately, $\pi_\theta(\mathbf{u}|\mathbf{x})$ cannot be computed directly, since the global policy is conditioned on the observation vector $\mathbf{o}$ and no knowledge about the observation distribution $P_t(\mathbf{o}|\mathbf{x})$ is assumed. However, as the full state $\mathbf{x}$ and the observation vector $\mathbf{o}_t$ are accessible at training time, a linearization of $\pi_\theta(\mathbf{u}|\mathbf{x})$ can be computed for each time step. Thus, points $\left(\mathbf{x}_t^i, \mathbb{E}_{\pi_\theta(\mathbf{u}|\mathbf{o}_t^i)}(\mathbf{u})\right)$ are taken from the current samples and used to approximate $\pi_\theta(\mathbf{u}|\mathbf{x})$ with a linear Gaussian distribution similar to the dynamics estimation. This estimated distribution is then used to compute the KL divergence. To reduce the required number of samples, a Gaussian mixture model can be used to obtain a prior based on current and previous samples. This is reasonable, as the global policy does not change much due to the step size $\epsilon$ imposed on the local trajectory optimization. Given the dynamics, the Lagrangian (13) can be solved efficiently by a standard Linear Quadratic Regulator (LQR) backward pass as described in Section 3.4.1.

### 3.4.1 Local Trajectory Optimization

This section contains the derivation of the LQR backward pass used in GPS to optimize the trajectories of the local controllers. The content of this section follows prior work [34]. Objective of local trajectory optimization is to optimize the augmented Lagrangian (13):

$$\mathcal{L}(\zeta) = \left( \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x},\mathbf{u})} \left[ \ell_t(\mathbf{x},\mathbf{u}) - \mathbf{u}^T \boldsymbol{\lambda}_t \right] + \upsilon_t \phi_t^{\zeta}(\zeta,\theta) \right) + \eta \left( D_{KL}\left( \zeta(\tau) \parallel \hat{\zeta}(\tau) \right) - \epsilon \right) \tag{13 revisited}$$

We start by expressing the first KL divergence term as en expectation over $\zeta_t(\mathbf{x},\mathbf{u})$:

$$\begin{aligned}
\phi_t^{\zeta}(\zeta,\theta) &= \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x})} \left[ D_{KL}\left( \zeta_t(\mathbf{u}|\mathbf{x}) \parallel \pi_\theta(\mathbf{u}|\mathbf{x}) \right) \right] \\
&= \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x})} \left[ \int_{-\infty}^{\infty} \zeta_t(\mathbf{u}|\mathbf{x}) \log \frac{\zeta_t(\mathbf{u}|\mathbf{x})}{\pi_\theta(\mathbf{u}|\mathbf{x})} d\mathbf{u} \right] \\
&= \iint_{-\infty}^{\infty} \zeta_t(\mathbf{x}) \zeta_t(\mathbf{u}|\mathbf{x}) \log \frac{\zeta_t(\mathbf{u}|\mathbf{x})}{\pi_\theta(\mathbf{u}|\mathbf{x})} d\mathbf{u} d\mathbf{x} \\
&= \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x},\mathbf{u})} \left[ \log \zeta_t(\mathbf{u}|\mathbf{x}) - \log \pi_\theta(\mathbf{u}|\mathbf{x}) \right]
\end{aligned}$$

(13) now becomes:

$$\mathcal{L}(\zeta) = \left( \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x},\mathbf{u})} \left[ \ell_t(\mathbf{x},\mathbf{u}) - \mathbf{u}^T \boldsymbol{\lambda}_t \right] + \upsilon_t \mathop{\mathbb{E}}_{\zeta_t(\mathbf{x},\mathbf{u})} \left[ \log \zeta_t(\mathbf{u}|\mathbf{x}) - \log \pi_\theta(\mathbf{u}|\mathbf{x}) \right] \right) + \eta \left( D_{KL}\left( \zeta(\tau) \parallel \hat{\zeta}(\tau) \right) - \epsilon \right) \tag{14}$$

Using the linearity of the expectation and defining

$$\begin{aligned}
c_t(\mathbf{x},\mathbf{u}) &:= \ell_t(\mathbf{x},\mathbf{u}) - \mathbf{u}^T \boldsymbol{\lambda}_t - \upsilon_t \log \pi_\theta(\mathbf{u}|\mathbf{x}) \\
c(\hat{\tau}) &:= \sum_{t=1}^{T} c_t(\mathbf{x},\mathbf{u})
\end{aligned}$$

(14) can be reformulated to

$$\mathcal{L}(\zeta) = \mathop{\mathbb{E}}_{\zeta(\tau)} \left[ c(\hat{\tau}) \right] - \upsilon_t \mathcal{H}(\zeta(\tau)) + \eta \left( D_{KL}\left( \zeta(\tau) \parallel \hat{\zeta}(\tau) \right) - \epsilon \right)$$

where $\mathcal{H}(\zeta(\tau)) = -\mathbb{E}_{\zeta(\tau)}[\log \zeta(\tau)]$ is the entropy of the local controller's trajectory distribution. Replacing the second KL divergence term with its definition then yields:

$$\begin{aligned}\mathcal{L}(\zeta) &= \mathop{\mathbb{E}}_{\zeta(\tau)}[c(\hat{\tau})] - \upsilon_t \mathcal{H}(\zeta(\tau)) + \eta \mathcal{H}(\zeta(\tau)) - \eta \mathop{\mathbb{E}}_{\zeta(\tau)}\left[\log \hat{\zeta}(\tau)\right] - \eta\epsilon \\ &= \mathop{\mathbb{E}}_{\zeta(\tau)}\left[c(\hat{\tau}) - \eta \log \hat{\zeta}(\tau)\right] + (\eta - \upsilon_t)\mathcal{H}(\zeta(\tau)) - \eta\epsilon\end{aligned}$$

The minimization of this Lagrangian can then be rewritten as:

$$\min_{\zeta(\tau)\in\mathcal{N}(\hat{\tau})} \mathop{\mathbb{E}}_{\zeta(\tau)}\left[\frac{1}{\eta+\upsilon_t} c(\hat{\tau}) - \frac{\eta}{\eta+\upsilon_t}\log \hat{\zeta}(\tau)\right] - \mathcal{H}(\zeta(\tau))$$

where $\mathcal{N}(\hat{\tau})$ is the set of Gaussian distributed trajectories. This objective corresponds to minimizing a maximum entropy problem:

$$\min_{\zeta(\tau)\in\mathcal{N}(\hat{\tau})} \mathop{\mathbb{E}}_{\zeta(\tau)}[\tilde{c}(\hat{\tau})] - \mathcal{H}(\zeta(\tau)) \text{ where} \qquad \tilde{c}(\hat{\tau}) = \frac{1}{\eta+\upsilon_t} c(\hat{\tau}) - \frac{\eta}{\eta+\upsilon_t}\log \hat{\zeta}(\tau)$$

The parameters $\mathbf{K}_t$, $\mathbf{k}_t$ and $\mathbf{\Sigma}_t$ of the optimal linear Gaussian controller $\zeta_t(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mathbf{K}_t\mathbf{x} + \mathbf{k}_t; \mathbf{\Sigma}_t)$ can now be computed with a standard LQR backward pass [57], which is summarized below.

Since the cost per step $\tilde{c}_t(\mathbf{x},\mathbf{u})$ are not quadratic, we use a quadratic approximation around the trajectory mean of the previous local controller $(\overline{\mathbf{x}}_1, \overline{\mathbf{u}}_1, \ldots, \overline{\mathbf{x}}_T, \overline{\mathbf{u}}_T) := \mathbb{E}_{\hat{\zeta}(\tau)}[\hat{\tau}]$:

$$\begin{aligned}\tilde{c}_t(\mathbf{x},\mathbf{u}) \approx \hat{c}_t(\mathbf{x},\mathbf{u}) &:= \frac{1}{2}\left[\mathbf{x}-\overline{\mathbf{x}}; \mathbf{u}-\overline{\mathbf{u}}\right]^T \tilde{c}_{t,\mathbf{xu},\mathbf{xu}}\left[\mathbf{x}-\overline{\mathbf{x}}; \mathbf{u}-\overline{\mathbf{u}}\right] + \left[\mathbf{x}-\overline{\mathbf{x}}; \mathbf{u}-\overline{\mathbf{u}}\right]^T \tilde{c}_{t,\mathbf{xu}} + const \\ &= \frac{1}{2}\left[\mathbf{x};\mathbf{u}\right]^T \tilde{c}_{t,\mathbf{xu},\mathbf{xu}}\left[\mathbf{x};\mathbf{u}\right] + \left[\mathbf{x};\mathbf{u}\right]^T \tilde{c}_{t,\mathbf{xu}} + const\end{aligned}$$

where subscripts denote derivatives, such that $\tilde{c}_{t,\mathbf{xu}}$ is the Jacobian of $\tilde{c}_t$ with respect to $[\mathbf{x};\mathbf{u}]$ and $\tilde{c}_{t,\mathbf{xu},\mathbf{xu}}$ is the Hessian, respectively.

The Q-function $Q_t(\mathbf{x},\mathbf{u})$ and the cost-to-go function $V_t(\mathbf{x})$ of this problem are generally defined as:

$$\begin{aligned}Q_t(\mathbf{x},\mathbf{u}) &= \hat{c}_t(\mathbf{x},\mathbf{u}) + V_{t+1}(\mathbf{f}_{\mathbf{x},t}\mathbf{x} + \mathbf{f}_{\mathbf{u},t}\mathbf{u} + \mathbf{f}_{c,t}) && \text{for } 1 \le t \le T-1 \\ Q_T(\mathbf{x},\mathbf{u}) &= \hat{c}_T(\mathbf{x},\mathbf{u}) \\ V_t(\mathbf{x}) &= \min_{\mathbf{u}} Q_t(\mathbf{x},\mathbf{u}) && \text{for } 1 \le t \le T\end{aligned}$$

Both functions $Q_t(\mathbf{x},\mathbf{u})$ and $V_t(\mathbf{x},\mathbf{u})$ are quadratic, which we will prove by recursively constructing their quadratic representations, starting at $T$. The assumption trivially holds for $Q_T(\mathbf{x},\mathbf{u})$, since it is equal to the quadratic cost function $\hat{c}_T(\mathbf{x},\mathbf{u})$. Assuming that $Q_t(\mathbf{x},\mathbf{u})$ is quadratic for any $1 \le t \le T$ enables us to write it in its quadratic form:

$$\begin{aligned}Q_t(\mathbf{x},\mathbf{u}) &= \frac{1}{2}\left[\mathbf{x};\mathbf{u}\right]^T Q_{t,\mathbf{xu},\mathbf{xu}}\left[\mathbf{x};\mathbf{u}\right] + \left[\mathbf{x};\mathbf{u}\right]^T Q_{t,\mathbf{xu}} + const \\ &= \frac{1}{2}\mathbf{x}^T Q_{t,\mathbf{x},\mathbf{x}}\mathbf{x} + \mathbf{u}^T Q_{t,\mathbf{u},\mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{u}^T Q_{t,\mathbf{u},\mathbf{u}}\mathbf{u} + \mathbf{x}^T Q_{t,\mathbf{x}} + \mathbf{u}^T Q_{t,\mathbf{u}} + const\end{aligned}$$

where $Q_{t,\mathbf{xu},\mathbf{xu}}$ is symmetric. To compute the coefficient matrices of $V_t(\mathbf{x})$, we compute the derivative of $Q_t(\mathbf{x},\mathbf{u})$ w.r.t. $\mathbf{u}$ and set it to zero:

$$\begin{aligned}0 &= Q_{t,\mathbf{u},\mathbf{u}}\mathbf{u} + Q_{t,\mathbf{u},\mathbf{x}}\mathbf{x} + Q_{t,\mathbf{u}} \\ \iff \mathbf{u} &= -Q_{t,\mathbf{u},\mathbf{u}}^{-1}Q_{t,\mathbf{u},\mathbf{x}}\mathbf{x} - Q_{t,\mathbf{u},\mathbf{u}}^{-1}Q_{t,\mathbf{u}}\end{aligned} \tag{15}$$

Using the assumed quadratic form of $Q_t(\mathbf{x},\mathbf{u})$, this gives:

$$\begin{aligned}V_t(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^T V_{t,\mathbf{x},\mathbf{x}}\mathbf{x} + \mathbf{x}^T V_{t,\mathbf{x}} + const \\ \text{where} \qquad V_{t,\mathbf{x},\mathbf{x}} &= Q_{t,\mathbf{x},\mathbf{x}} - Q_{t,\mathbf{u},\mathbf{x}}^T Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u},\mathbf{x}} \\ V_{t,\mathbf{x}} &= Q_{t,\mathbf{x}} - Q_{t,\mathbf{u},\mathbf{x}}^T Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u}}\end{aligned}$$

which implies that the Q-function of the previous time step is also a quadratic.

$$Q_{t-1}(\mathbf{x},\mathbf{u}) = \frac{1}{2}[\mathbf{x};\mathbf{u}]^T Q_{t-1,\mathbf{xu},\mathbf{xu}}[\mathbf{x};\mathbf{u}] + [\mathbf{x};\mathbf{u}]^T Q_{t-1,\mathbf{xu}} + const$$
$$\text{where} \quad Q_{t-1,\mathbf{xu},\mathbf{xu}} = \tilde{c}_{t-1,\mathbf{xu},\mathbf{xu}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t,\mathbf{x},\mathbf{x}} \mathbf{f}_{\mathbf{xu},t}$$
$$Q_{t-1,\mathbf{xu}} = \tilde{c}_{t-1,\mathbf{xu}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t,\mathbf{x}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t,\mathbf{x},\mathbf{x}} \mathbf{f}_{c,t}$$

Note that the symmetry of $Q_{t-1,\mathbf{xu},\mathbf{xu}}$ follows directly from the symmetry of $V_{t,\mathbf{x},\mathbf{x}}$. Thus, the Q-functions and value functions of all time steps are quadratic and their coefficients can be computed with the following recurrence, starting at $t = T$:

$$Q_{t,\mathbf{xu},\mathbf{xu}} = \tilde{c}_{t,\mathbf{xu},\mathbf{xu}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t+1,\mathbf{x},\mathbf{x}} \mathbf{f}_{\mathbf{xu},t}$$
$$Q_{t,\mathbf{xu}} = \tilde{c}_{t,\mathbf{xu}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t+1,\mathbf{x}} + \mathbf{f}_{\mathbf{xu},t}^T V_{t+1,\mathbf{x},\mathbf{x}} \mathbf{f}_{c,t}$$
$$V_{t,\mathbf{x},\mathbf{x}} = Q_{t,\mathbf{x},\mathbf{x}} - Q_{t,\mathbf{u},\mathbf{x}}^T Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u},\mathbf{x}}$$
$$V_{t,\mathbf{x}} = Q_{t,\mathbf{x}} - Q_{t,\mathbf{u},\mathbf{x}}^T Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u}}$$

Using equation (15), the optimal control law is given by:

$$g(\mathbf{x}) = \mathbf{K}_t \mathbf{x} + \mathbf{k}_t$$
$$\text{where} \quad \mathbf{K}_t = Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u},\mathbf{x}}$$
$$\mathbf{k}_t = Q_{t,\mathbf{u},\mathbf{u}}^{-1} Q_{t,\mathbf{u}}$$

To optimize the maximum entropy objective, the controller's covariance $\mathbf{\Sigma}_t$ must be set to $Q_{t,\mathbf{u},\mathbf{u}}^{-1}$, as shown in prior work [51]. Consequently, the optimal linear-Gaussian controller is $\mathcal{N}\left(\mathbf{K}_t\mathbf{x} + \mathbf{k}_t; Q_{t,\mathbf{u},\mathbf{u}}^{-1}\right)$.

### 3.5 Supervised Global Policy Optimization

Objective of global policy optimization is to optimize $\pi_\theta(\mathbf{u}|\mathbf{o})$ such that the KL divergence between $\pi_\theta(\mathbf{u}|\mathbf{x})$ and the local controllers $\zeta_t^i(\mathbf{u}|\mathbf{x})$ and the expectation of the Lagrangian term $\mathbf{u}^T\boldsymbol{\lambda}_{i,t}$ becomes minimal:

$$\mathcal{L}_\theta(\theta,\zeta) = \sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\pi_\theta(\mathbf{u}|\mathbf{x})\zeta_t(\mathbf{x})}\left[\mathbf{u}^T\boldsymbol{\lambda}_{i,t}\right] + \upsilon_t \phi_t^\theta(\theta,\zeta)$$

Unlike the local controllers, the global policy is conditioned on the observation vector $\mathbf{o}$ only. Thus, knowledge about the full system state $\mathbf{x}$ is not required at test time. Since $\pi_\theta(\mathbf{u}|\mathbf{o})$ is Gaussian distributed, the global policy can be represented by two functions $\mu^\pi(\mathbf{o})$ and $\Sigma^\pi(\mathbf{o})$, such that $\pi_\theta(\mathbf{u}|\mathbf{o}) = \mathcal{N}(\mu^\pi(\mathbf{o}), \Sigma^\pi(\mathbf{o}))$. Using the definition of the KL divergence, this yields:

$$\begin{aligned}\mathcal{L}_\theta(\theta,\zeta) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T} \upsilon_t \mathop{\mathbb{E}}_{P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})}\Big[&tr\left[\mathbf{C}_{t,i}^{-1}\Sigma^\pi(\mathbf{o})\right] - \log|\Sigma^\pi(\mathbf{o})| \\ &+ \left(\mu^\pi(\mathbf{o}) - \mu_{t,i}^\zeta(\mathbf{x})\right)^T \mathbf{C}_{t,i}^{-1}\left(\mu^\pi(\mathbf{o}) - \mu_{t,i}^\zeta(\mathbf{x})\right) + \frac{2}{\upsilon_t}\boldsymbol{\lambda}_{i,t}^T\mu^\pi(\mathbf{o})\Big]\end{aligned} \tag{16}$$

where $\mu_{t,i}^\zeta(\mathbf{x})$ and $\mathbf{C}_{t,i}$ are mean and covariance of the local controller: $\zeta_t^i(\mathbf{u}|\mathbf{x}) = \mathcal{N}\left(\mu_{t,i}^\zeta(\mathbf{x}), \mathbf{C}_{t,i}\right)$. The expectation under $P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})$ is evaluated using samples from the real system, which were drawn in the beginning of the iteration. Additionally to the state, the observations $\mathbf{o}$ belonging to the states $\mathbf{x}$ need to be recorded at sampling time, since the observation distribution $P_t(\mathbf{o}|\mathbf{x})$ is unknown. Since $\mu^\pi(\mathbf{o})$ and $\Sigma^\pi(\mathbf{o})$ are independent functions, they can be optimized individually, as described in the following two sections.

#### 3.5.1 Global Policy Mean Optimization

When optimizing (16) w.r.t. $\mu^\pi(\mathbf{o})$, all summands independent of $\mu^\pi(\mathbf{o})$ can be removed and the objective becomes:

$$\min_{\mu^\pi} \frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})}\left[\upsilon_t\left(\mu^\pi(\mathbf{o}) - \mu_{t,i}^\zeta(\mathbf{x})\right)^T \mathbf{C}_{t,i}^{-1}\left(\mu^\pi(\mathbf{o}) - \mu_{t,i}^\zeta(\mathbf{x})\right) + 2\boldsymbol{\lambda}_{i,t}^T\mu^\pi(\mathbf{o})\right]$$

which is equivalent to

$$\min_{\mu^\pi} \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})} \Bigg[ \upsilon_t \left( \mu^\pi(\mathbf{o}) - \mu^\zeta_{t,i}(\mathbf{x}) + \frac{1}{\upsilon_t} \mathbf{C}_{t,i} \boldsymbol{\lambda}_{i,t} \right)^T \mathbf{C}_{t,i}^{-1} \left( \mu^\pi(\mathbf{o}) - \mu^\zeta_{t,i}(\mathbf{x}) + \frac{1}{\upsilon_t} \mathbf{C}_{t,i} \boldsymbol{\lambda}_{i,t} \right) + 2\boldsymbol{\lambda}_{i,t}^T \mu^\zeta_{t,i}(\mathbf{x}) + \frac{1}{\upsilon_t} \boldsymbol{\lambda}_{i,t}^T \mathbf{C}_{t,i} \boldsymbol{\lambda}_{i,t} \Bigg]$$

Dropping again terms independent of $\mu^\pi(\mathbf{o})$ gives:

$$\min_{\mu^\pi} \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})} \left[ \upsilon_t \left( \mu^\pi(\mathbf{o}) - \mu^\zeta_{t,i}(\mathbf{x}) + \frac{1}{\upsilon_t} \mathbf{C}_{t,i} \boldsymbol{\lambda}_{i,t} \right)^T \mathbf{C}_{t,i}^{-1} \left( \mu^\pi(\mathbf{o}) - \mu^\zeta_{t,i}(\mathbf{x}) + \frac{1}{\upsilon_t} \mathbf{C}_{t,i} \boldsymbol{\lambda}_{i,t} \right) \right]$$

Thus, the objective becomes an Euclidean loss function between the means of global policy and local controller, offset by the Lagrangian term and weighted by the local controller's precision matrix. In this work, $\mu^\pi(\mathbf{o})$ is represented by a fully connected neural network. The training data of this network is a set of observation-action pairs $(\mathbf{o}_t, \mathbf{u}_t)$. To generate these pairs, observation state pairs $(\mathbf{o}_t, \mathbf{x}_t)$ are obtained from the samples drawn at the beginning of the iteration and the controller mean of the respective optimized local controllers is computed: $\left(\mathbf{o}_t, \mathbb{E}_{\zeta_t(\mathbf{u}|\mathbf{x})}[\mathbf{u}]\right)$. Due to the definition of the local controllers, the mean is computed as $\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$.

### 3.5.2 Global Policy Covariance Optimization

When optimizing $\Sigma^\pi(\mathbf{o})$, objective (16) becomes:

$$\min_{\Sigma^\pi} \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} \upsilon_t \mathop{\mathbb{E}}_{P_t(\mathbf{o}|\mathbf{x})\zeta_t(\mathbf{x})} \left[ tr\left[ \mathbf{C}_{t,i}^{-1} \Sigma^\pi(\mathbf{o}) \right] - \log |\Sigma^\pi(\mathbf{o})| \right]$$

as all other summands are independent of $\Sigma^\pi(\mathbf{o})$. In this work, $\Sigma^\pi$ was chosen not to depend on the observation, which yields the following objective:

$$\min_{\Sigma^\pi} \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} \upsilon_t \left( tr\left[ \mathbf{C}_{t,i}^{-1} \Sigma^\pi \right] - \log |\Sigma^\pi| \right)$$

Differentiating and setting to zero gives:

$$0 = \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} \upsilon_t \left( \mathbf{C}_{t,i}^{-1} - (\Sigma^\pi)^{-1} \right)$$

which results in the following equation for $\Sigma^\pi$:

$$\Sigma^\pi = \left( \sum_{i=1}^{N} \sum_{t=1}^{T} \upsilon_t \right) \left( \sum_{i=1}^{N} \sum_{t=1}^{T} \upsilon_t \mathbf{C}_{t,i}^{-1} \right)^{-1}$$

## 4 Experimental Procedure

In this section we describe the experiments we conducted to validate GPS performance for in-hand manipulation. The aim of the experiments is to examine whether GPS is capable of acquiring in-hand manipulation skills from scratch within a reasonable amount of time. Hence, we designed three experiments that require different levels of hand dexterity. In the first experiment, GPS has to learn to reach and hold a defined hand pose from a range of starting poses, while compensating for gravity. This experiment does not include any object manipulation and was conducted to evaluate whether GPS is generally capable of dealing with the high-dimensionality of the Allegro hand. The second experiment is a task in which a fixed cylinder has to be rolled between two fingers. Since the position of the cylinder varies, the tactile sensors have to be utilized to to determine the rolling strategy to be used. This experiment was deployed to analyze GPS capability of dealing with non-linear contact dynamics and performing in-hand manipulation on a fixed object. To evaluate GPS performance on freely-moving objects, we conducted a third experiment, in which a grasped cuboid has to be moved to a 3D target location. This experiment requires substantially more dexterity than the previous experiment, since the cuboid is not fixed and can thus slip and drop if not handled carefully. The objective of this experiment is to evaluate how GPS is performing on one of the key challenges of dexterous manipulation - in-hand object reconfiguration. In the following, we describe the platform we used and provide details to the different experimental setups. The results of the experiments are collectively described in section 5.

### 4.1 Setup

All our experiments run in simulation, which enables us to perform rapid testing while still giving a reasonable insight into our method's performance. We use a simulated version of the Allegro Hand [35] with tactile sensors on the fingertips. The general design of the Allegro Hand is similar to the human hand, although it has 4 fingers instead of 5. Each finger consists of 4 fully controllable revolute joints, including one joint able to rotate the entire finger around its longitudinal axis, while the other joints are placed similar to the joints of a human hand. The simulated tactile sensors measure orthogonal external force on 23 points around each fingertip. Since these sensors are placed on each finger tip, they generate a 92 dimensional continuous input signal, which serves as only external input to the final policy. Combined with the joint positions and velocities, the hand provides a 124 dimensional input signal. As all joints are torque-controlled, the resulting continuous control space is 16-dimensional.

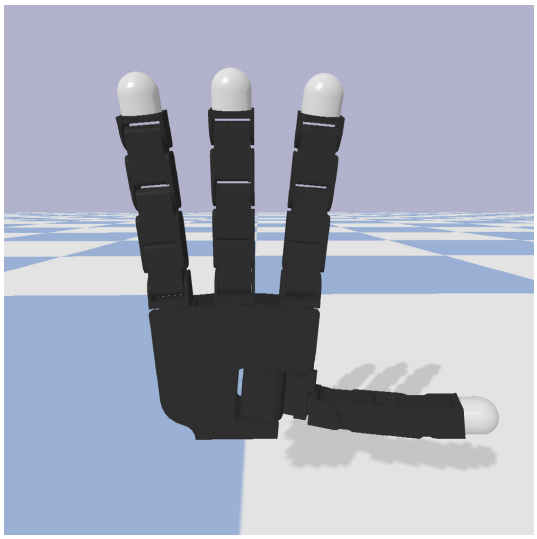

**Figure 3:** Allegro hand in simulation.

We use `pybullet` as simulation platform. As the name suggests, `pybullet` is a `python` API for the `Bullet 3` physics engine. We configured `pybullet` to use a simulation time step of 1ms to ensure that the contact dynamics are simulated with sufficient precision. Every 10ms, the current hand state is fed into the policy and a new control signal is generated. The length of a roll-out is 1s in all our experiments, which corresponds to a trajectory of 100 states. In every iteration, we draw 30 samples for each initial state, which corresponds to 30s robot time per initial state. Except for the second experiment, gravity is generally enabled in all simulations to make the scenarios as realistic as possible. We chose the global policy to be represented by a fully-connected neural network with 6 hidden layers and 150 rectified linear units (ReLU) in each layer.

### 4.2 Positioning Fingers

In this experiment, we evaluate the general ability of GPS to control complex high-dimensional systems like the Allegro hand. In order to do so, we provide a constant target hand configuration (see Figure 4) and learn a controller to reach and hold this configuration starting from a range of initial configurations. Here, a hand configuration is defined by the positions of all joints of the hand, joint velocities are ignored. Since gravity is simulated in this experiment, the task includes to learn gravity compensation additionally to the positioning control task. While this task is generally solved from an engineering perspective, it gives valuable insight into the ability of GPS to estimate a dynamics model of the hand and use it for a standard control task.

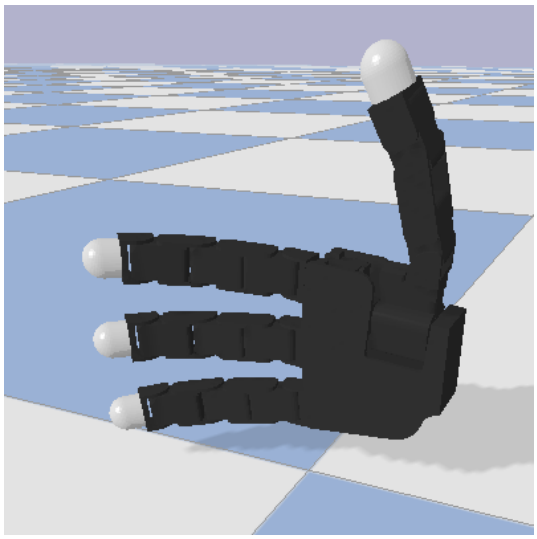

**Figure 4:** Target hand configuration for the finger positioning task. The objective is to reach this configuration and hold it until the end of the roll-out.

We chose the number of initial states for training to be 4. The initial states correspond to different hand configurations, which are depicted in Figure 5. To generate these configurations, we sample from a Gaussian distribution around the target joint angles. Since no object is manipulated in this experiment, we disable the tactile sensors to reduce the input dimensions. Consequently, the joint positions and velocities serve as only inputs to the local and global policies, which results in a 32 dimensional input space. We use the summed squared distance to the target joint positions as cost function and additionally penalize torques to ensure smooth and efficient trajectories:

$$\ell_t(\mathbf{x}, \mathbf{u}) = \|\mathbf{u}\|^2 + \|pos_{joint}(\mathbf{x}) - p_{target}\|^2$$

where $p_{target}$ is the vector of target joint positions and $pos_{joint} : \mathbb{R}^{32} \rightarrow \mathbb{R}^{16}$ extracts the joint positions from the state.

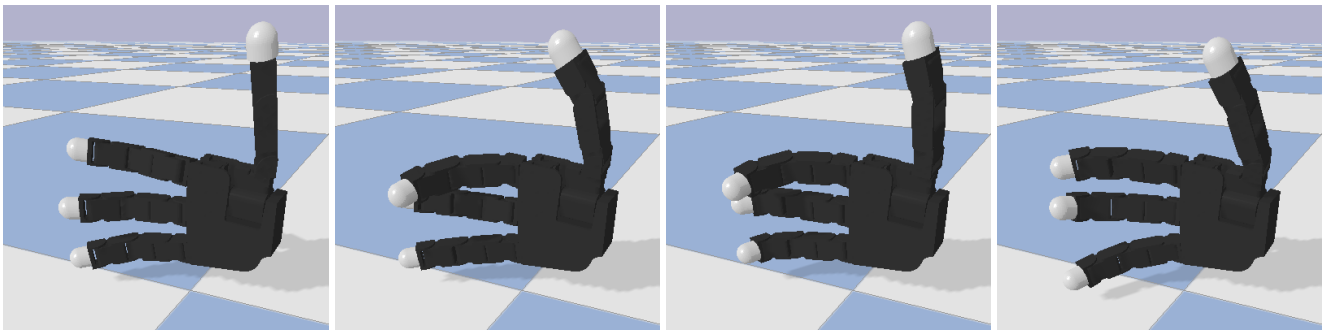

**Figure 5:** Initial hand configurations used to train local controllers for the finger positioning task.

## 4.3 Rolling a Fixed Cylinder

While the previous experiment gives insight into the ability of GPS to control the Allegro hand, it does not touch manipulation and the challenges related to it. Additionally to the inherent higher complexity of the task, these challenges include underactuation, more complex dynamics and the partial observability of the state. The underactuation occurs naturally when manipulating objects, since the object pose cannot be controlled directly but has to be influenced by the fingers. Furthermore, occurrence of contact introduces non-linearities to the dynamics, which makes their estimation harder. In the previous experiment, the dynamics of the fingers were mutually independent as long as no contact occurred. When manipulating an object, however, this is not the case anymore, as all fingers in contact with the object can influence each other. This makes the system dynamics significantly more complex and introduces the necessity to coordinate finger movement, contrary to the previous experiment.

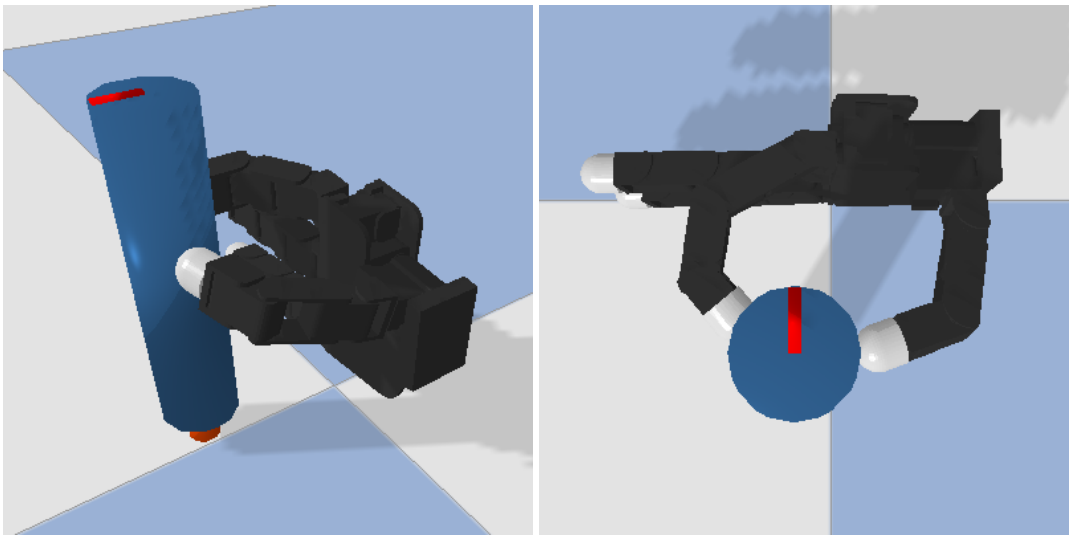

**Figure 6:** Experimental setup of the cylinder rolling task. The left picture shows the system in the initial configuration. The objective is to roll the cylinder by exactly 180° using thumb and index finger, reaching the configuration depicted in the right image. The location of the cylinder on the plane differs for each initial state.

In this experiment, we test the ability of GPS in dealing with the challenges stated above by introducing an in-hand manipulation task. The objective of this task is to rotate a grasped cylinder 180° as depicted in Figure 6. The cylinder is fixed on the ground and can only be rotated around its longitudinal axis. Here, the initial states correspond to different positions of the cylinder on the ground plane as shown in Figure 7. To ensure that they are feasible, we draw these positions from a uniform distribution in a range of 6cm × 1cm, instead of drawing them from a Gaussian distribution.

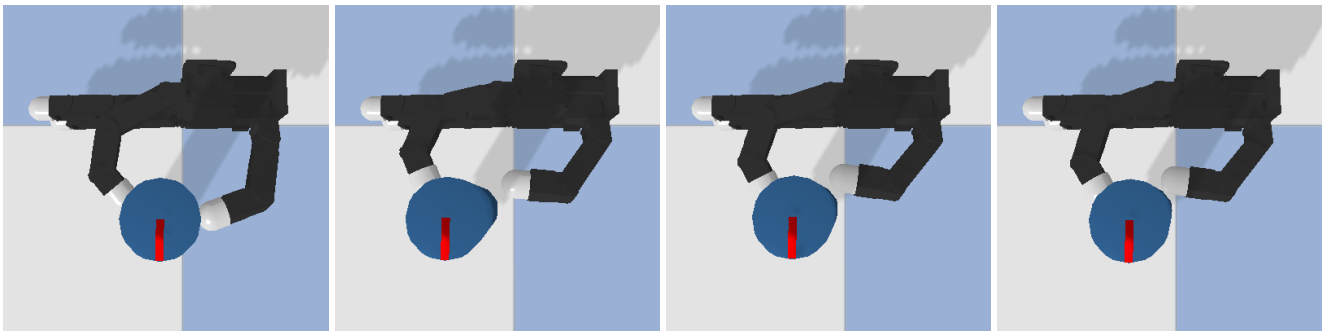

**Figure 7:** Initial configurations used to train local controllers for the cylinder rolling task. The cylinder locations on the plane are selected randomly from a two dimensional uniform distribution.

We chose the number of different initial training configurations to be 4. The small finger and the ring finger as well as the twist joints of each finger are disabled, since they provide no advantage in solving this task. As the policy has no access to any past observations and the task requires to lose contact with the cylinder, we provide the policy with the position of the cylinder. This compensates for the fact that the policy has no way of knowing the cylinder location while it is not in contact with any of the tactile sensors. Considering the 23 dimensional input of the tactile sensors for each finger, this results in a 60 dimensional input space and a 6 dimensional output space for the final policy. For local optimization, however, we do not include tactile sensor information of any kind, since we found it to slow down convergence due to the increased complexity of the state. Instead, we include the cylinder angle and velocity, which results in a 14 dimensional state. Note that the cylinder position is constant for each local controller and thus not beneficial to add. Performance is measured using using the following cost function:

$$\ell_t(\mathbf{x}, \mathbf{u}) = \|\mathbf{u}\|^2 + \min\left(\left(angle_{cylinder}(\mathbf{x}) \pm \pi\right)^2\right)$$

where $angle_{cylinder} : \mathbb{R}^{54} \rightarrow \mathbb{R}$ extracts the cylinder angle from the state. Similar to the previous experiment, we penalize control commands to ensure a smooth and efficient trajectory.

The previous experiment evaluates GPS on the manipulation of a fixed object using two fingers. While that task covers the ability of GPS to perform manipulation in general, solving it does not require the full capability of the Allegro hand. Hence, in this experiment we take a step further and evaluate GPS's ability to reorient freely-moving objects in-hand. Contrary to the previous experiment, we enable all fingers and joints in order to leave the robot as unrestricted as possible. Additionally to the higher state and control dimensions, this task introduces further new challenges, since the movement of the manipulated object is not restricted anymore. This essentially shifts the movement range from a 1 dimensional space to 6 dimensions. Furthermore, the object can now be dropped and thus the robot has to learn to perform a stable grasp prior to any manipulation attempts.

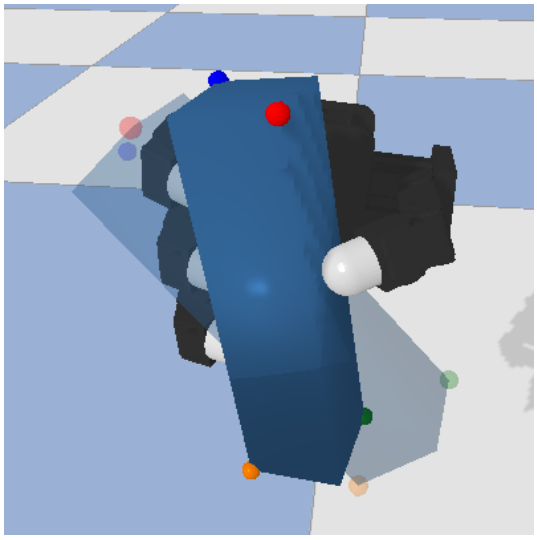

**Figure 8:** Experimental setup of the cuboid positioning task. The objective is to reconfigure the cuboid from its initial pose to the target pose marked by the transparent cuboid. Note that the hand is 0.5m above the ground.

Objective of this task is to reconfigure a grasped cuboid into a constant target pose as shown in Figure 8. Since the hand is controlled in torque control mode, the initial grasp is not stable and the object will drop if no action is taken. In this scenario, the initial states correspond to different poses of the cuboid. Similar to the previous experiment, the poses are drawn from a uniform distribution to ensure that they are feasible. The ranges of the uniform distribution compared to the target pose can be obtained from Table 1.

| | Sampling range | Target value |
|---|---|---|
| x | 0.1 ±0.05m | 0.1 m |
| y | 0.0 ±0.01m | 0.0 m |
| z | 0.0m | 0.0 m |
| roll | 0.0 ±11.5 ° | 45.0 ° |
| pitch | 0.0 ±1.7 ° | 0.0 ° |
| yaw | 11.3 ±1.7 ° | 11.3 ° |

**Table 1:** Initial cuboid pose sampling range. Each initial state is generated by drawing all parameters from a uniform random distribution with the respective range.

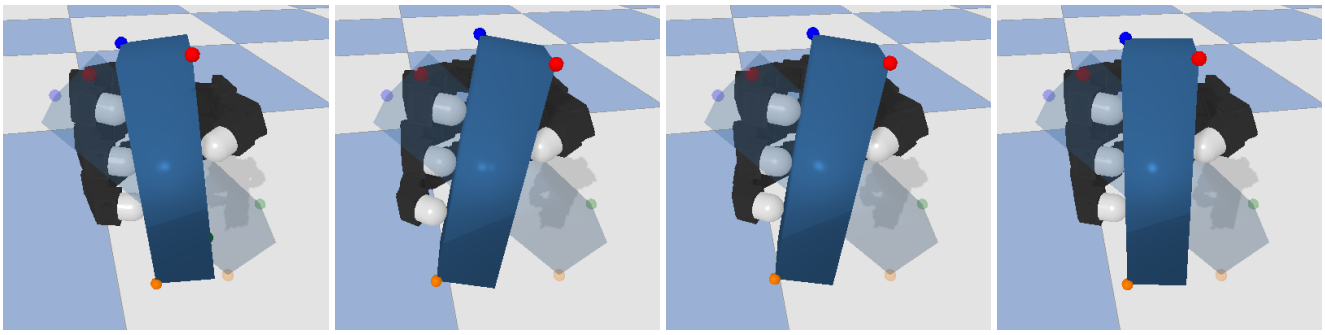

**Figure 9:** Initial configurations used to train local controllers for the cuboid positioning task. For each initial state, the cuboid's pose is drawn from a uniform random distribution, further specified in Table 1.

Similar to the previous experiments, we use 4 distinct initial states for training. As all joints and fingers are enabled, this results in a 124 dimensional input space (16 joint positions, 16 joint velocities, 92 tactile forces) and 16 dimensional output space for the final policy. For the local controller optimization we drop the tactile forces again and replace them with the positions and linear velocities of 4 markers placed on the cuboid. As visible in Figure 9, the markers are placed on opposite corners on both sides of the cuboid and their position, hence, fully defines the pose of the cuboid. Using the marker positions to encode the object pose instead of using object position and orientation directly has the advantage that the distance between two poses can be measured in terms of a euclidean distance. Alternatively, the orientation offset could also be measured by computing the error angle to the target pose. However, this requires trigonometric functions and thus leads to a more complicated cost term. Additionally, we found our measure to produce better results. Jointly with the joint state, this results in a 56 dimensional state space for the local optimization. Utilizing this distance measure, the cost function used for this experiment is:

$$\ell_t(\mathbf{x}, \mathbf{u}) = \|\mathbf{u}\|^2 + d(\mathbf{x}) + 10^{-3} \log\left(d(\mathbf{x}) + 10^{-6}\right)$$

where $d(\mathbf{x}) = \sum_{i=1}^{4} \|m_i(\mathbf{x}) - p_i\|^2$ is the summed squared marker distance, $p_i \in \mathbb{R}^3$ is the target position of marker $i$ and $m_i : \mathbb{R}^{56} \to \mathbb{R}^3$ extracts the position of the $i$-th marker from the state. This type of cost function has been proposed for tasks which require high precision in prior work [13]. The idea behind this is that the quadratic term encourages the optimization to quickly converge towards the target state, even from a far distance. As soon as the distance becomes small, however, the quadratic term will become very shallow and less helpful for the optimization. Adding the logarithmic term includes a concave "spike" around 0 to the cost function, which leads to a quicker convergence for small distances.

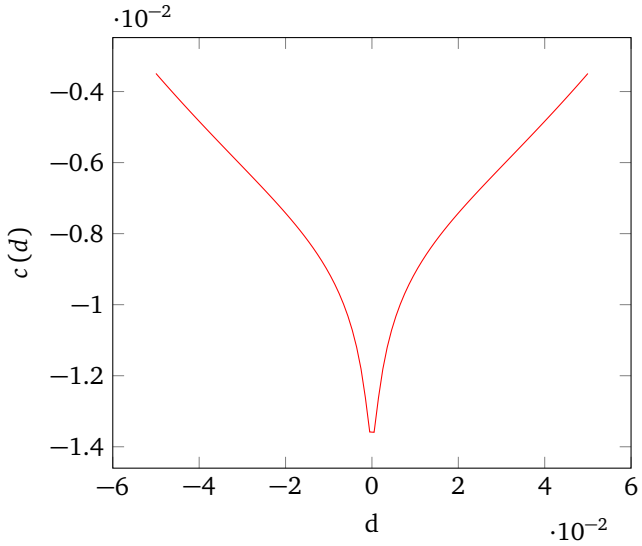


**Figure 10:** Plot of the distance term used in the cost function: $c(d) = d^2 + 10^{-3} \log\left(d^2 + 10^{-6}\right)$. The concave shape introduced by the logarithmic term encourages the optimization to reach the target position as precisely as possible.

## 5 Results

In this section we summarize the results of the experiments we conducted. For each task, we compare the performances of the local controllers and the global policy over the course of a single GPS execution. Additionally to the 4 training configurations, the global policies were tested on 100 randomly generated unseen initial configurations in every iteration. This validates that the global policies did not overfit but solved their respective tasks in a general way. To enable a comparison, the performances of the global policy on seen and unseen configurations are visualized individually in every task.

Since the probability of success shall be maximized at test time, the control signal with the highest probability is chosen at each time step during the roll-outs presented here. As the local controllers as well as the global policy are conditionally Gaussian, the control signal with the highest probability is the controller mean in both cases. Considering that the simulator is deterministic, roll-outs of a controller on a fixed initial configuration will always result in the same trajectory. Thus, the presented results are generated by single roll-outs of the local controller and the global policy on each initial configuration in each iteration of each experiment.

A video of the global policy performing roll-outs on all three tasks can be found here: `https://youtu.be/b0IGw4WI60k`.

### 5.1 Positioning Fingers

Figure 11 shows the learning progress of GPS in the finger positioning task. Since the value of the cost function does not provide an intuitive explanation of the performance, the mean joint position error at the final time step is added for comparison. As shown in Figure 11, the policy has been trained for 30 iterations which corresponds to 1h of robot time. The final policy was not only able to deliver good performance on seen configurations, but also achieved a similar performance on unseen configurations (refer to Figure 12 for examples of unseen configurations). This indicates that the policy did not overfit, and indeed solved the problem generally. Hence, these results show that GPS is capable of dealing with high-dimensional systems like the Allegro hand.

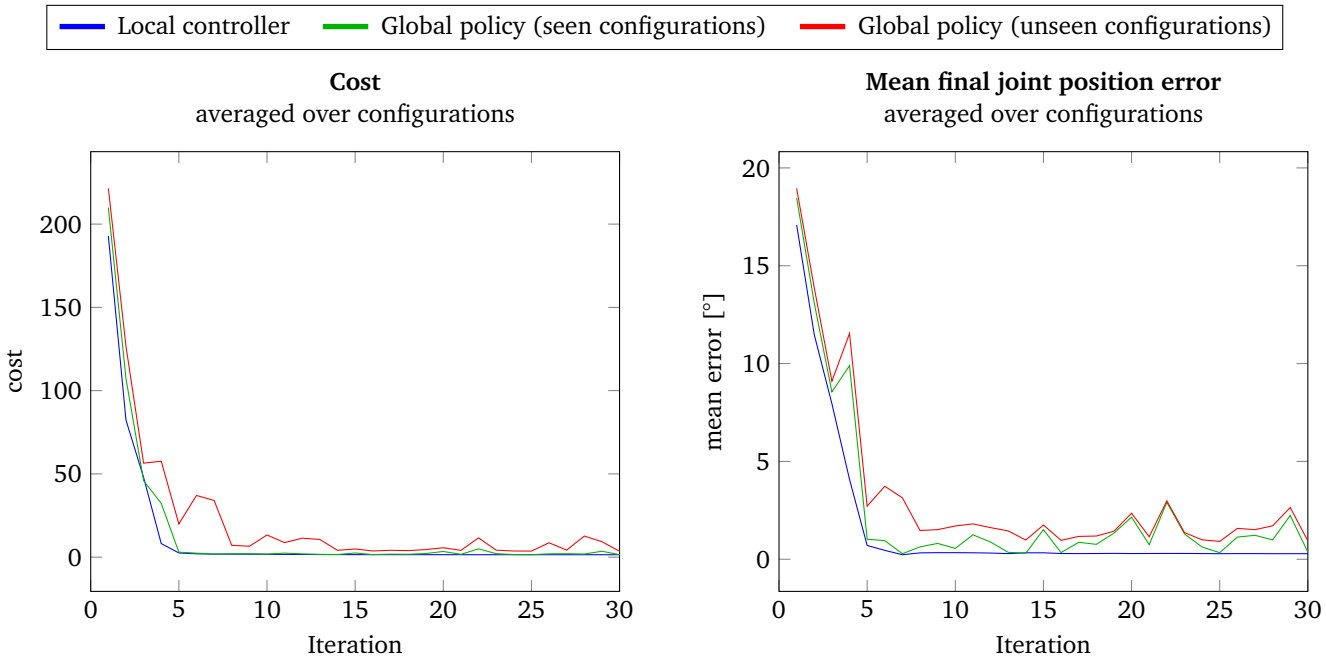


**Figure 11:** Performance comparison of the local controllers and the global policy over the course of one GPS execution on the finger positioning task. The mean final joint position error is defined as mean position error of all joints of the Allegro hand at the final time step of the respective roll-out. In the last iteration, the average mean final joint position errors are 0.28° for the local controller, 0.35° for the global policy on seen states and 0.89° on unseen states.

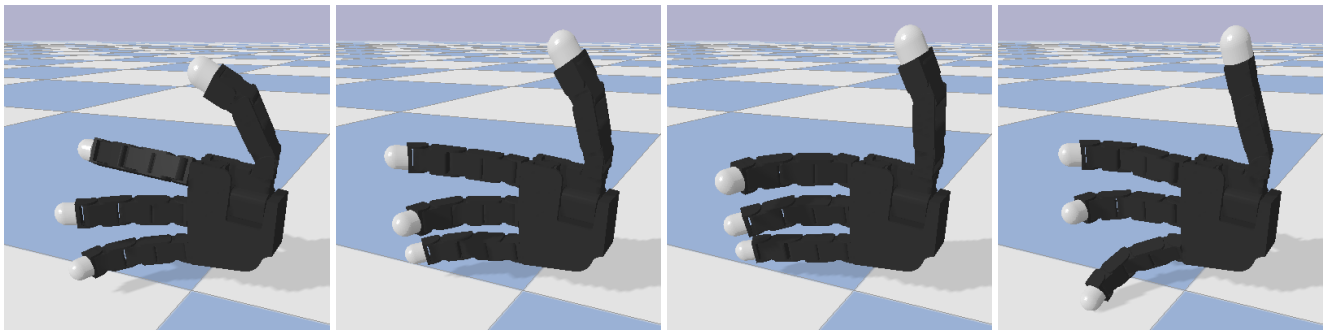

**Figure 12:** Examples of unseen initial hand configurations used for policy testing in the finger positioning task.

## 5.2 Rolling a Fixed Cylinder

Figure 13 depicts the learning progress of GPS in the cylinder rolling task. The course of the cylinder angle error in the final time step of each roll-out is visualized to provide an intuitive understanding of the optimization progress. In this task, the training was performed for 50 iterations, which corresponds to 1:40h of robot time. Both the local controllers and the global policy achieved reasonable improvement over the iterations of this experiment. As in the previous experiment, the global policy did not overfit and was able to deliver similar performance on seen and unseen configurations (refer to Figure 14 for examples of unseen configurations).

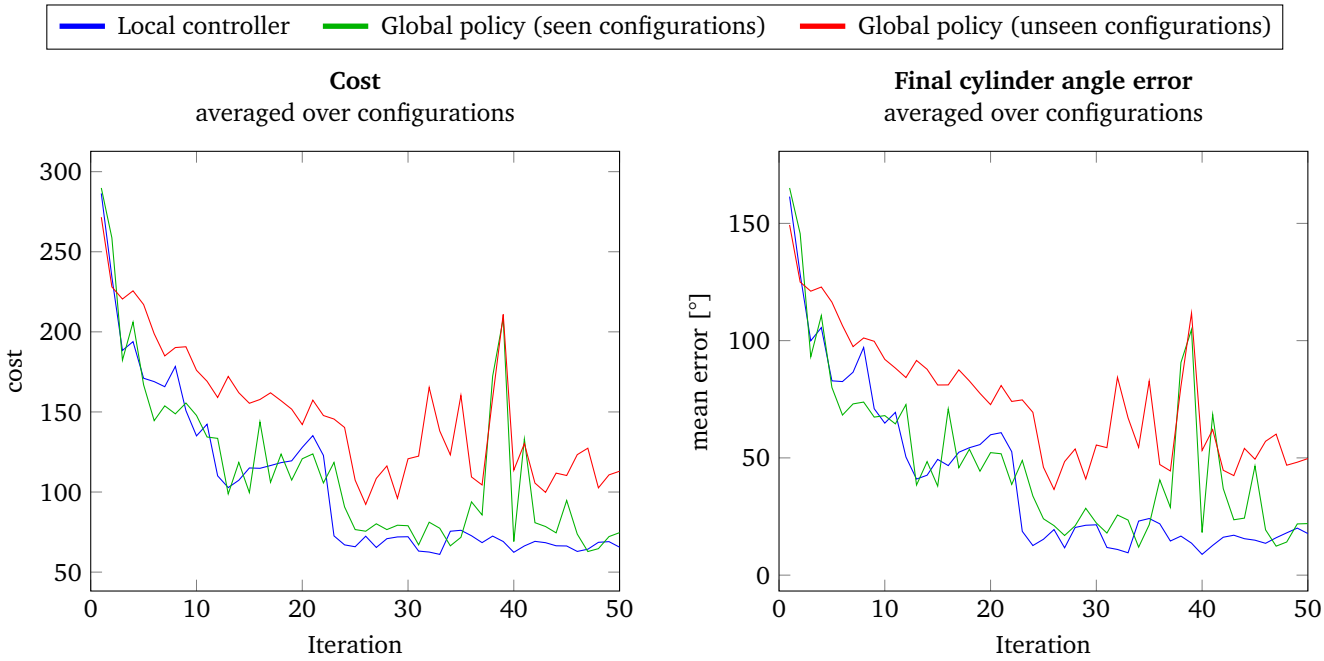


**Figure 13:** Performance comparison of the local controllers and the global policy over the course of one GPS execution on the cylinder rolling task. The final cylinder angle error is defined as the difference between the cylinder angle and the target angle of 180° at the final time step of the respective roll-out. In the last iteration, the average final angle errors are 17.73° for the local controller, 21.98° for the global policy on seen configurations and 49.71° on unseen configurations.

However, as visible in the graphs, the local optimization converged towards an error of approximately 35° and not to 0°. The error occurs most likely due to the fact that the robot cannot perform a 180° rotation with one rolling motion of the finger. Thus, in order to achieve the target angle, the robot has essentially two options: perform a finger-gait or flick the cylinder to its target position. Since the optimization method used here is restricted by a KL-divergence to the previous trajectory, it does not perform much exploration and focuses on exploitation instead. To find strategies like finger-gaiting, however, exploration is required, since during a “step” of the finger no progress is made and the gradient of the cost function is 0. Hence, we found that the local controller optimization typically develops flicking strategies that require a precise amount of force but little planning-ahead to minimize the error angle. For most of the tested

configurations, the flicking strategies aim to place the cylinder in the right angle with a single flick without any further intervention. An example of a roll-out on such a configuration is shown in Figure 15. In the other tested configurations, the cylinder is intentionally flicked with enough force to overshoot, but then stopped with the index finger and rolled to the target orientation as shown in Figure 16. This strategy typically results in lower overall cost, since the cylinder is spun faster and thus the error angle is reduced quicker as in the other strategy. Recall, that the cost are not only computed for the final time step but accumulated over all time steps of the roll-out.

Although flicking can be perfected to minimize the error, it is very prone to control errors and the cylinder tends to overshoot or undershoot as visible in Figure 15. Small perturbations in the fingertip position quickly make the difference between a good flick or one that either misses the cylinder or hits the cylinder at an unfortunate angle. Also when using the second strategy, as the cylinder is perfectly round, the robot cannot measure its orientation when stopping it and thus has to rely solely on the precision of the initial flick. This high-accuracy requirement is especially challenging for the global policy, which has to adapt the flicking strategy for unseen cylinder positions. The error-sensitivity of this strategy is most likely also the cause for the jagged cost graph of the global policy.

**Figure 14:** Examples of unseen initial configurations used for policy testing in the cylinder rotation task.

Nevertheless, these results show clearly that GPS is capable of dealing with complex discontinuous dynamics and is able to perform in-hand manipulation. The presented strategies were all learned from scratch and GPS achieved significant improvement within 50 iterations.

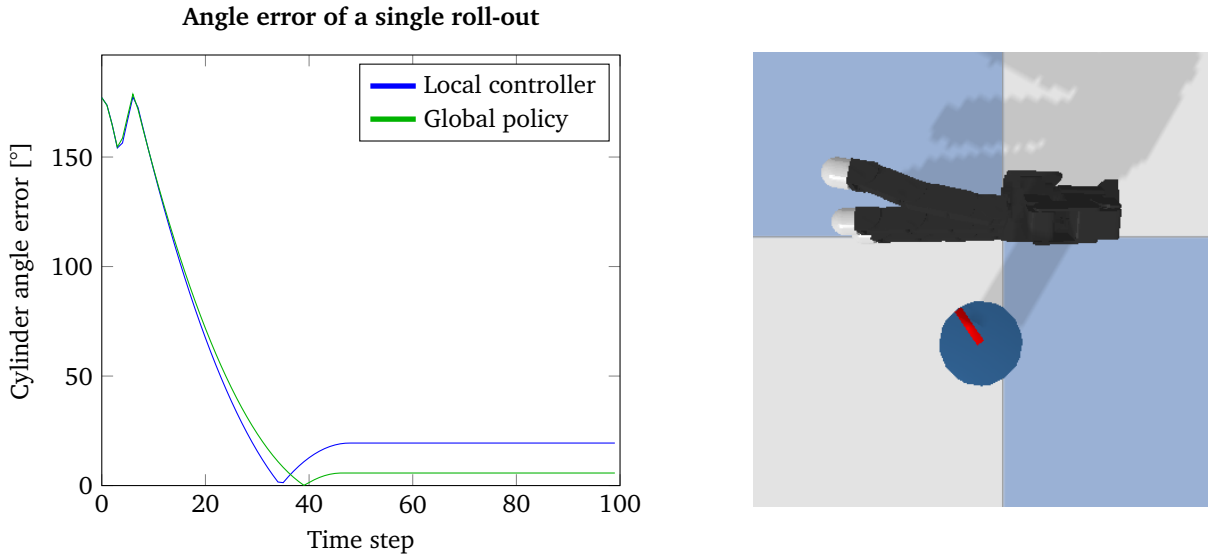


**Figure 15:** Comparison of the angle error over a single roll-out of the global policy and the respective local controller in iteration 50. Both controllers performed a flicking motion with the index finger at the beginning and did not touch the cylinder again. The cylinder overshot slightly in both cases. On the right is a picture of the final time step of the global policy roll-out.

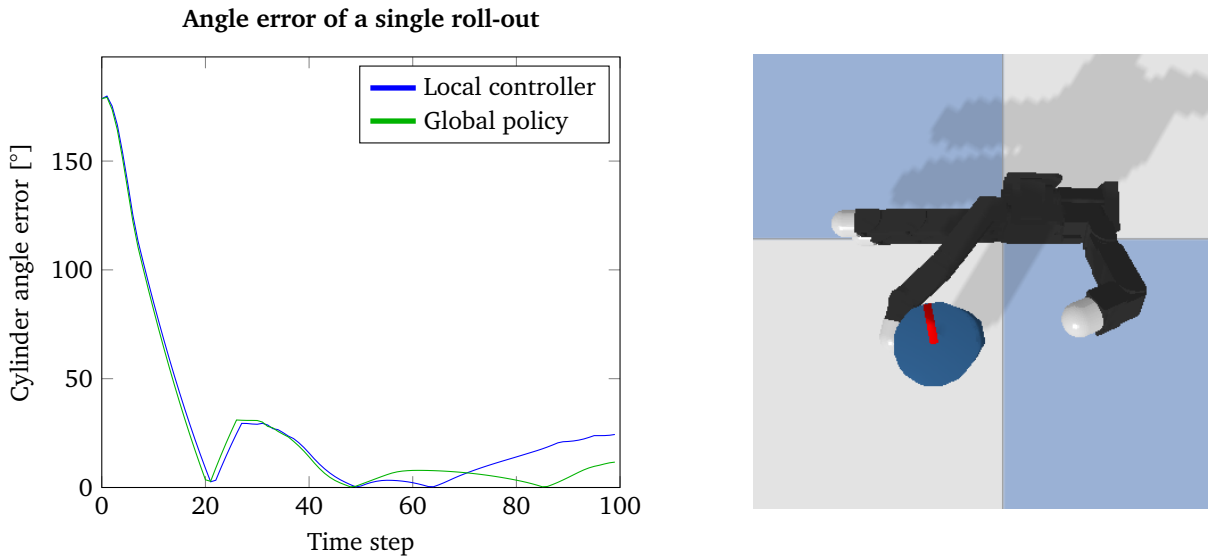


**Figure 16:** Comparison of the angle error over another roll-out of the global policy and the respective local controller in iteration 50. Both controllers performed a strong flick and corrected the orientation of the cylinder after it overshot. On the right is a picture of the final time step of the global policy roll-out.

## 5.3 Reconfiguring a Freely-Moving Cuboid

Figure 18 shows the learning progress of GPS in the cuboid positioning task. To provide an intuition about the optimization progress, we visualize further performance measures additionally to the cost. These performance measures consist of the mean marker position error and the error in cuboid position and orientation at the final time step of the roll-outs.

At the beginning of the optimization, GPS has to learn to stabilize the object first to avoid dropping it. Since the floor is 0.5m below the hand, the inflicted cost upon dropping are fairly high, which explains the high cost at the beginning and the spikes throughout the optimization. After the object has been stabilized, the algorithm continues to optimize the cuboid's pose and is finally capable of reaching the target pose with high accuracy as visible in Figure 19. As shown in Figure 18, the policy generalizes and achieves similar performance on seen and unseen configurations (refer to Figure 17 for examples of unseen configurations). The reason for the temporarily bad performance on seen configurations is that GPS struggles to find a stable grasp for one of the four training configurations and thus often drops the object. However, in iteration 80, a stable grasp for this configuration has been acquired.

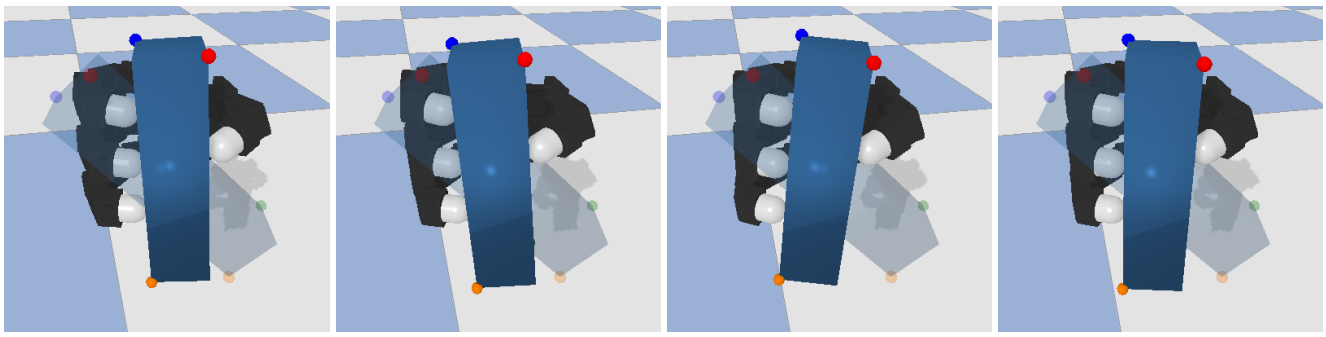

**Figure 17:** Examples of unseen initial configurations used for policy testing in the cuboid positioning task.

These results show that our method is capable of performing dexterous manipulation of a freely-moving object. The task introduces significant challenges, since the object can slip and drop and is inherently more complex than the previous two tasks due to the higher degrees of freedom. Despite these challenges, an accurate and robust policy was learnt in 80 iterations, which corresponds to 2h and 40min of robot time.

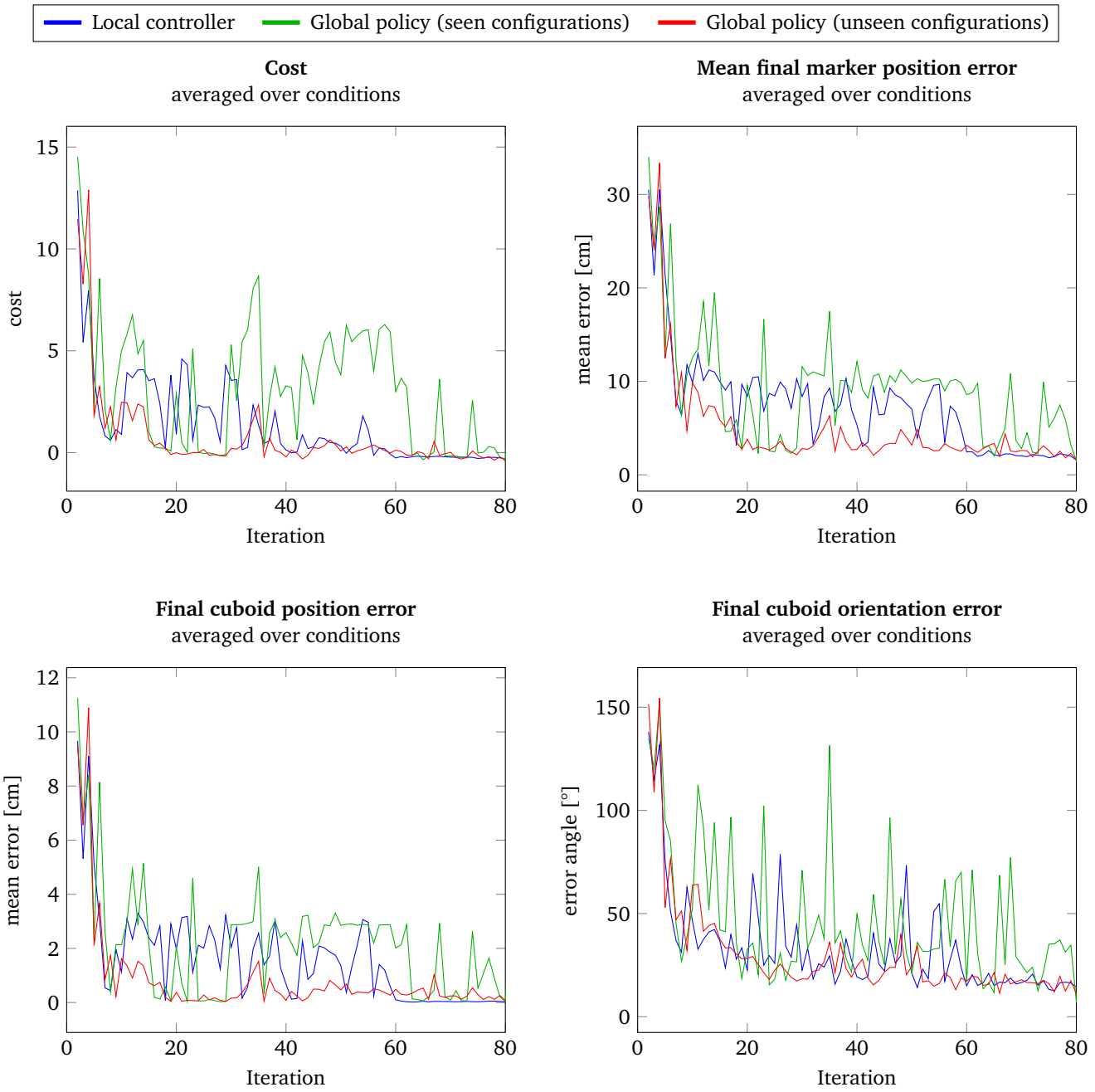


**Figure 18:** Performance comparison of the local controllers and the global policy over the course of the optimization for the cuboid positioning task. The mean marker position error is computed as the mean position error over all 4 markers. The cuboid position error is computed as the distance of the cuboid center to its target location, while the orientation error is measured as the angle between current and target orientation. All performance measures except for the cost are computed at the final time step of the respective roll-out. In the last iteration, the average final marker errors are 2.0cm for the local controller and 3.1cm for the policy on seen configurations and 2.3cm for the policy on unseen configurations.

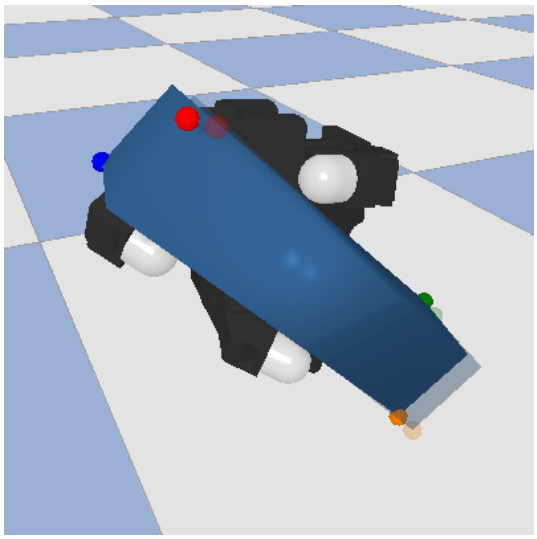

**Figure 19:** Final state of the global policy applied to an unseen configuration. All fingers are used to stabilize the object in a very accurate position and orientation. The general configuration of the hand looks natural despite the fact that it was not trained on human demonstrations.

## 6 Conclusion

In this work, we used Guided Policy Search to learn policies for in-hand manipulation driven on tactile feedback. We demonstrated that our method is capable of performing simple control tasks, like position control for the fingers of a robot hand, but also presented results for two in-hand manipulation tasks. One task consists of the manipulation of a fixed object with two fingers, while the other task's objective is to reposition a freely-moving object in-hand using all fingers. The algorithm is not provided with any domain or robot specific knowledge and merely uses a high-level cost function which defines the objective of the task. Especially the lack of a robot model is challenging, since complex high-dimensional dynamics have to be estimated from roll-outs on the robot. Despite these challenges, our method was able to achieve reasonable performance for distinct initial states and managed to learn a global neural network policy for each presented task. We validated that the global policies did not overfit, by testing each of them on 100 randomly generated unseen states. Our results show that the algorithm is capable of learning dexterous in-hand manipulation policies that solely rely on tactile sensor data and the internal robot state as input.

An issue of this work is that it does not use the original, but a modified BADMM version to deal with probability density functions instead of finite dimensional vectors as optimization variables. While Levine et al. [34] claim that the modified version inherits the proven convergence guarantees of the original BADMM, they do not provide a proof. Experimental evaluation suggests that the method converges at a good rate, but without a mathematical proof it remains unclear if this holds for all cases. Especially since the convergence bounds of BADMM depend on the problem dimensionality, a mathematical analysis of the convergence of the modified BADMM would add confidence in this method. However, since performing such an analysis is beyond the scope of this thesis, this question remains open.

Following up to this thesis, a couple of issues could be addressed in future work. While simulated results already give a reasonable insight into the method's performance, it has to be tested on real hardware to confirm its applicability in the real world. This certainly comes with a variety of new challenges, since the real world cannot be simply reset to some initial state. Thus, tasks need to be found, in which the robot can either reset itself after each roll-out or in which a human can do this with sufficient precision and at a reasonable time effort. While the cylinder task falls in the first category, our last experiment is probably not executable in the real world, since the cuboid would drop before the robot could take any action. One option is to alter this experiment by placing the cuboid on a solid surface or by finding a different intrinsically stable initial configuration similar to Kumar et al. [18].

Furthermore, it would be interesting to see if, given more time and initial states for training, the algorithm is capable of producing a global policy that solves our final task for a broader range of initial configurations. These initial configurations could, for example, include a higher range of initial object poses or maybe even different grasps or deviations in the object size or shape. This would obviously introduce a completely new challenge to the training procedure, since for each of these configurations or objects, a proper grasp needs to be found prior to training, which is an entire research topic on its own. However, solving the problem of in-hand object reconfiguration in a more general way would certainly mean a big step towards general hand-based dexterity.

## References


[1] Richard W. Young. Evolution of the human hand: the role of throwing and clubbing. *J Anat*, 202(1):165–174, Jan 2003. ISSN 0021-8782. doi: 10.1046/j.1469-7580.2003.00144.x. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1571064/. 12587931[pmid].

[2] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7, June 2011. doi: 10.1109/ICAR.2011.6088576.

[3] Yunfei Bai and C. Karen Liu. Dexterous manipulation using both palm and fingers. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565, 2014.

[4] Antonio Bicchi, Raffaele Sorrentino, et al. Dexterous manipulation through rolling. In *ICRA*, volume 95, pages 452–457, 1995.

[5] Li Han and Jeffrey C Trinkle. Dextrous manipulation by rolling and finger gaiting. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 730–735. IEEE, 1998.

[6] Moëz Cherif and Kamal K Gupta. Planning quasi-static fingertip manipulations for reconfiguring objects. *IEEE Transactions on Robotics and Automation*, 15(5):837–848, 1999.

[7] Ryuta Ozawa, Suguru Arimoto, Shinsuke Nakamura, and Ji-Hun Bae. Control of an object with parallel surfaces by a pair of finger robots without object sensing. *IEEE Transactions on Robotics*, 21(5):965–976, 2005.

[8] Zoe Doulgeri and Leonidas Droukas. On rolling contact motion by robotic fingers via prescribed performance control. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3976–3981. IEEE, 2013.

[9] Katharina Hertkorn, Maximo A Roa, and Christoph Borst. Planning in-hand object manipulation with multifingered hands considering task constraints. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 617–624. IEEE, 2013.

[10] Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6215–6222. IEEE, 2015.

[11] Yiannis Karayiannidis, Karl Pauwels, Christian Smith, Danica Kragic, et al. In-hand manipulation using gravity and controlled slip. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5636–5641. IEEE, 2015.

[12] Nikhil Chavan Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. *CoRR*, abs/1707.00318, 2017. URL http://arxiv.org/abs/1707.00318.

[13] Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. *CoRR*, abs/1501.05611, 2015. URL http://arxiv.org/abs/1501.05611.

[14] Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1503–1510. IEEE, 2015.

[15] Ivaylo Popov, Nicolas Heess, Timothy P. Lillicrap, Roland Hafner, Gabriel Barth - Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin A. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *CoRR*, abs/1704.03073, 2017. URL http://arxiv.org/abs/1704.03073.

[16] H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, volume 38, page 995–1003. JMLR, 2015. URL http://jmlr.org/proceedings/papers/v38/vanhoof15.pdf.

[17] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017. URL http://arxiv.org/abs/1709.10087.

[18] Vikash Kumar, Abhishek Gupta, Emanuel Todorov, and Sergey Levine. Learning dexterous manipulation policies from experience and imitation. *CoRR*, abs/1611.05095, 2016. URL http://arxiv.org/abs/1611.05095.

[19] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3786–3793. IEEE, 2016.

[20] Raphael Deimel and Oliver Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *The International Journal of Robotics Research*, 35(1-3):161–185, 2016.

[21] Leif P Jentoft, Qian Wan, and Robert D Howe. Limits to compliance and the role of tactile sensing in grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6394–6399. IEEE, 2014.

[22] Yasemin Bekiroglu, Janne Laaksonen, Jimmy Alison Jorgensen, Ville Kyrki, and Danica Kragic. Assessing grasp stability based on learning and haptic data. *IEEE Transactions on Robotics*, 27(3):616–629, 2011.

[23] Jonna Laaksonen, Ekaterina Nikandrova, and Ville Kyrki. Probabilistic sensor-based grasping. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2019–2026. IEEE, 2012.

[24] Hao Dang and Peter K Allen. Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 36(4):309–330, 2014.

[25] Joseph M Romano, Kaijen Hsiao, G ü nter Niemeyer, Sachin Chitta, and Katherine J Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079, 2011.

[26] Taro Takahashi, Toshimitsu Tsuboi, Takeo Kishida, Yasunori Kawanami, Satoru Shimizu, Masatsugu Iribe, Tetsuharu Fukushima, and Masahiro Fujita. Adaptive grasping by multi fingered hand with tactile sensor based on robust force and position control. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 264–271. IEEE, 2008.

[27] Miao Li, Hang Yin, Kenji Tahara, and Aude Billard. Learning object-level impedance control for robust grasping and dexterous manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6784–6791. IEEE, 2014.

[28] Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E Gil Jones. Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1228–1235. IEEE, 2010.

[29] Eric L Sauser, Brenna D Argall, Giorgio Metta, and Aude G Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 60(1):55–71, 2012.

[30] Hitoshi Maekawa, Kazuo Tanie, and Kiyoshi Komoriya. Tactile sensor based manipulation of an unknown object by a multifingered hand with rolling contact. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 743–750. IEEE, 1995.

[31] Hong Zhang and Ning Nicholas Chen. Control of contact via tactile sensing. *IEEE Transactions on Robotics and Automation*, 16(5):482–495, 2000.

[32] Benjamin Ward-Cherrier, Nicolas Rojas, and Nathan F Lepora. Model-free precise in-hand manipulation with a 3d-printed tactile gripper. *IEEE Robotics and Automation Letters*, 2(4):2056–2063, 2017.

[33] Yevgen Chebotar, Oliver Kroemer, and Jan Peters. Learning robot tactile sensing for object manipulation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3368–3375. IEEE, 2014.

[34] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL `http://jmlr.org/papers/v17/15-522.html`.

[35] Wonik Robotics. Allegro hand. `http://www.simlab.co.kr/Allegro-Hand.htm`. Acessed: 2018-09-20.

[36] T. Erez, S. Kolev, and E. Todorov. Receding-horizon online optimization for dexterous object manipulation. In *IEEE Conference on Robotics and Automation (ICRA)*, 2014.

[37] E. Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306 vol. 1, June 2005. doi: 10.1109/ACC.2005.1469949.

[38] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144. Eurographics Association, 2012.

[39] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185539. URL http://doi.acm.org/10.1145/2185520.2185539.

[40] P. J. Sanz, A. P. Del Pobil, J. M. Inesta, and G. Recatala. Vision-guided grasping of unknown objects for service robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 4, pages 3018–3025 vol.4, May 1998. doi: 10.1109/ROBOT.1998.680889.

[41] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008. doi: 10.1177/0278364907087172. URL https://doi.org/10.1177/0278364907087172.

[42] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM, 2007.

[43] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202, 2008.

[44] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[45] Urbain Prieur, Véronique Perdereau, and Alexandre Bernardino. Modeling and planning high-level in-hand manipulation actions from human knowledge and active learning from demonstration. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1330–1336. IEEE, 2012.

[46] Freek Stulp, Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning to grasp under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5703–5708. IEEE, 2011.

[47] Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, and Stefan Schaal. Learning force control policies for compliant manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4639–4644. IEEE, 2011.

[48] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning policy improvements with path integrals. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 828–835, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/theodorou10a.html.

[49] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 1607–1612. AAAI Press, 2010. URL http://dl.acm.org/citation.cfm?id=2898607.2898863.

[50] Vikash Kumar, Abhishek Gupta, Emanuel Todorov, and Sergey Levine. Learning dexterous manipulation policies from experience and imitation. *CoRR*, abs/1611.05095, 2016. URL http://arxiv.org/abs/1611.05095.

[51] Sergey Levine and Vladlen Koltun. Guided policy search. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1–9, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/levine13.html.

[52] Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 207–215. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5178-variational-policy-search-via-trajectory-optimization.pdf.

[53] Sergey Levine and Vladlen Koltun. Learning complex neural network policies with trajectory optimization. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 829–837, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/levine14.html.

[54] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1071–1079. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5444-learning-neural-network-policies-with-guided-policy-search-under-unknown-dynamics.pdf.

[55] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011. URL http://www.jmlr.org/proceedings/papers/v15/ross11a/ross11a.pdf.

[56] Huahua Wang and Arindam Banerjee. Bregman alternating direction method of multipliers. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2816–2824. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5612-bregman-alternating-direction-method-of-multipliers.pdf.

[57] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems., 01 2004.