Learning Hierarchical Control for Robust In-Hand Manipulation

Tingguang Li[1,2], Krishnan Srinivasan[2], Max Qing-Hu Meng[1], Wenzhen Yuan[3] and Jeannette Bohg[2]

Abstract— **Robotic in-hand manipulation has been a long-standing challenge due to the complexity of modelling hand and object in contact and of coordinating finger motion for complex manipulation sequences. To address these challenges, the majority of prior work has either focused on model-based, low-level controllers or on model-free deep reinforcement learning that each have their own limitations. We propose a hierarchical method that relies on traditional, model-based controllers on the low-level and learned policies on the mid-level. The low-level controllers can robustly execute different manipulation primitives (reposing, sliding, flipping). The mid-level policy orchestrates these primitives. We extensively evaluate our approach in simulation with a 3-fingered hand that controls three degrees of freedom of elongated objects. We show that our approach can move objects between almost all the possible poses in the workspace while keeping them firmly grasped. We also show that our approach is robust to inaccuracies in the object models and to observation noise. Finally, we show how our approach generalizes to objects of other shapes.**

I. INTRODUCTION

Dexterous Manipulation refers to the ability of changing the pose of an object to any other pose within the workspace of a hand [1–3]. In this paper, we are particularly concerned with the ability of in-hand manipulation where the object is continuously moved within the hand without dropping. This ability is used frequently in human manipulation e.g. when grasping a tool and re-adjusting it within the hand, when inspecting an object, when assembling objects or when adjusting an unstable grasp. Yet, in-hand manipulation remains a long-standing challenge in robotics despite the availability of multi-fingered dexterous hands such as [4–6].

Controlling dexterous manipulation can be divided into three levels [1]: (i) High-level control for task and motion planning as well as grasp selection, (ii) mid-level control for smoothly coordinating and sequencing manipulation phases towards a goal pose and (iii) low-level control that tracks the reference signal provided by the mid-level control with e.g. force or impedance controllers. The low-level has received considerable attention in prior work [1, 2, 7–10]. These methods assume some prior knowledge of the geometry and inertia of hand and object as well as of contact locations. They can robustly control local adjustments of the object pose in-hand. However, mid-level control is required to

[1]T. Li and M. Meng are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China. {tgli, Max}@ee.cuhk.edu.hk

[2]T. Li, K. Srinivasan and J. Bohg are with the Computer Science Department, Stanford University, USA. {tgli, krshna, bohg}@stanford.edu

[3]W. Yuan is with the Robotics Institute, Carnegie Mellon University, USA. yuanwz@cmu.edu
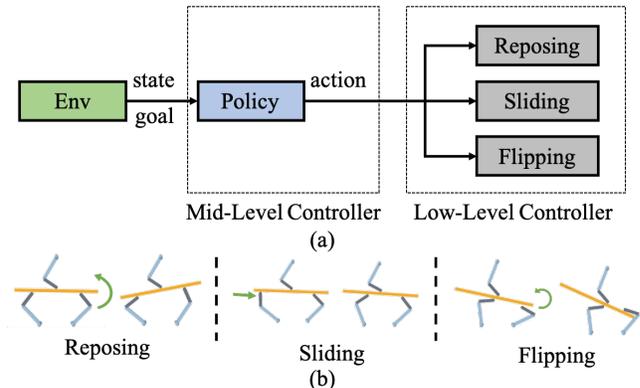
Fig. 1: (a) The proposed two-level hierarchical architecture for in-hand manipulation. At the low level, we have three manipulation primitives that are executed with low-level, torque controllers. At the mid-level, we train a policy with DRL to pick a manipulation primitive and its parameterization. (b) Visualization of the three manipulation primitives.

sequence manipulation phases and thereby to reach more distant object poses. Compared to low and high-level control, mid-level control is not as well studied [1, 11].

More recently, *Deep Reinforcement Learning* (DRL) has been used to learn dexterous manipulation. These are typically end-to-end approaches that do not assume a division between the low- and mid-level [12–14]. While these approaches do not require prior knowledge of hand and object properties, policies are learned per object instance. Also, these works are typically demonstrated on manipulation tasks that do not require the object to be continuously held by the hand. For example, the palm may always be facing upwards to prevent the object from dropping [12, 13], or the hand is manipulating an articulated object that is rigidly attached in the world [14]. Learning to hold the object firmly while reposing would likely require DRL methods an even larger amount of training episodes and a very carefully designed reward function. Nevertheless, in many real-world manipulation tasks this is an important ability.

We are proposing a hierarchical control structure for in-hand manipulation of objects within the entire reachable workspace. At the low-level, we use well-studied, compliant torque controllers per manipulation primitive. At the mid-level, we use DRL to learn a policy that sequences manipulation primitives towards a more distant object goal pose. Through this combination, we ensure that the object is continuously held by the hand. While the low-level controllers require prior knowledge about hand and object properties, we show that our framework is robust against observation noise in the object pose and inaccuracies of kinematic and dynamic models and thereby generalizes to object variations.

We define three manipulation primitives: reposing, sliding,

and flipping, as shown in Fig. 2. Reposing, or in-grasp manipulation [15], refers to moving the object while keeping the same contact configuration. Sliding and flipping change the contact point of one finger by either sliding it along the surface or moving it to a different side of the object. Different contact configurations have different ranges of poses that the object can reach. We train a policy with DRL to first reach a feasible contact configuration of the object and then select among these three manipulation primitives to repose the object to the goal pose. We perform an extensive evaluation of our approach in a simulated environment consisting of a three-fingered hand that can change the 3DoF pose (2D position and orientation) of pole-shaped objects. We also show that the overall hierarchical approach is suitable for other object shapes.

Compared to a graph search-based approach [16] and an end-to-end baseline, our method has a significantly higher success rate of reaching a target object pose for a pole, especially when they are far from the initial poses and complex finger coordination is required.

II. Related Work

A. Model-based Approaches to Dexterous Manipulation

Traditional approaches towards dexterous manipulation typically assume known kinematics, dynamics and contact models. They devise controllers that execute primitives such as reposing/in-grasp manipulation, finger sliding or finger gaiting. Rus [7, 9] developed different methods of rotating an object in a plane, involving a force-direction closure grasp. Okada [17], Kerr and Roth [18] applied sliding and rolling of robot fingers to manipulate objects of various shapes using inverse kinematics. A typical challenge for model-based approaches is to reach more distant object poses that requires longer manipulation sequences which switch between different primitives. To address this and plan a sequence of primitives, early work by [19] proposed a finger gaiting scheme based on geometrical modeling. Mordatch et al. [20] developed a trajectory optimization method that uses a contact-invariant cost. Fan et al. [21] developed a two-staged planner that first plans finger gaits and then uses a controller to move the object. However, neither of these methods has been demonstrated to be robust to noise and inaccurate models. Odhner and Dollar [22], Calli and Dollar [23] propose a method to roll or slide objects in-hand. Their method uses model-predictive control within a visual-servoing scheme. The authors specifically focus on underactuated hands with a low DoF while we consider high-DOF, fully-actuated hands. Platt et al. [24] define several conceptual states to describe the space of grasp configurations and explore the manipulation behavior in the context of a Markov Decision Process. Simpkins and Todorov [25] developed a hierarchical control scheme that can plan ideal forces and torques. Cruciani et al. [16], Murooka et al. [26] formulated the manipulation problem as a graph and generated the manipulation sequence by searching for a path in that graph. These approaches typically make strong assumptions like sticky contacts with infinite friction [25],

fully-known, deterministic transition models [16, 26] and a finite state space [24], which make them less likely to cope with real world challenges.

B. Model-Free Approaches to Dexterous Manipulation

Different from traditional methods, model-free approaches towards dexterous manipulation use limited or no prior knowledge of the gripper or object. They learn to control the object through reinforcement learning typically without distinguishing between the low and mid-level. These end-to-end learning approaches learn a mapping between observations of the object and control commands of the robot hand. Kumar et al. [12] developed a sample-efficient method for training an anthropomorphic hand to rotate a cylinder. At the core, they optimize linear-gaussian controllers via iterative-LQR. Andrychowicz et al. [13] train object-specific DRL policies for in-hand reposing. Van Hoof et al. [27] train a policy that uses tactile sensory data as input. The robot learns to roll and reorient cylindrical objects on a table. The policy is robust to changes in the size and weight of the objects. However, neither work requires the robot to robustly hold the object while manipulating it as the object is typically supported by either the upward-facing palm of the hand, a table or it is an articulated object fixed to the environment.

Furthermore, model-free DRL has a high sample complexity. To address this, [14, 28] combined imitation learning and reinforcement learning. However, the considered tasks are also constructed such that the object does not need to be grasped continuously and does not easily drop outside the reach of the hand. Learning a policy from scratch that firmly holds an object during in-hand manipulation may require excruciatingly high number of episodes and carefully tuned reward function.

C. Combining Deep Reinforcement Learning With Models

We propose a hybrid framework to combine the advantages of both, traditional model-based and more recent model-free approaches. To this end, Silver et al. [29], Johannink et al. [30] propose to learn residual policies that add corrective joint-commands to an existing controller. Thereby they account for model inaccuracies and noise. Our method is hierarchical in structure where the DRL policy selects manipulation primitives to be executed by the low-level controller, as opposed to being compositional, where the policy and controller output are summed.

III. Method

We propose a method to perform in-hand manipulation of objects with a fully-actuated hand, i.e. moving an object from pose X_0 to the target pose X_g. Specifically, we consider the challenging problem where the object has to be held continuously to be prevented from dropping out of reach. We model the manipulation state with (X, C) where X is the object pose relative to the hand, and C is the contact configuration of the fingers relative to the object frame. Our approach is based on the insight that this manipulation state space has a specific structure (see Fig. 2).
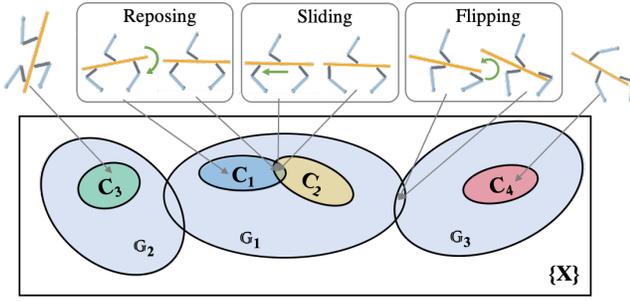
Fig. 2: A schematic of the object pose space $\{X\}$, contact configurations C, grasp postures $\mathbb{G}$ and motion primitives. Each contact configuration C covers a range of object poses X into which the object can be moved by the hand via *Reposing*. Different Cs may have some overlapping areas in object pose space such that *Sliding* or *Flipping* can be used to transfer between them. The in-hand manipulation process can be considered as finding a path between two points in $\{X\}$, while different motion primitives are needed to move within or between Cs.

In that space, one object pose can be realized by multiple contact configurations. One contact configuration can move the object to a range of poses. However, some object pose changes may require changing the contact configuration by either sliding on the object or flipping one finger to the other side of the object. Moreover, flipping results in a different grasp posture $\mathbb{G}$ where one finger makes new contact with a different side of the object. For this example task, we only assume hard-finger contacts [31] at the fingertips and precision grasps. To navigate the structure in this manipulation state space, we design three manipulation primitives that either *repose* the object without changing the contact configuration, *slide* a finger on the object or *flip* a finger to another side of the object. The last two primitives change the contact configuration and thereby the reachable object poses.

In this section, we give details on the low-level controllers that implement these three manipulation primitives and on the mid-level policies that learned to select from these primitives to move the object from the current to the goal state. This hierarchical approach allows the robot to navigate the complex manipulation state space and is visualized in Fig. 1 (a). While the overall idea is agnostic to the specific manipulation task, we use pole manipulation as a running example and show later how the idea generalizes to differently shaped objects. Throughout, we assume full observability of the system state and prior knowledge of inertia parameters of the hand and object. In the real-world, the object and hand could be visually tracked [32]. Furthermore, having access to approximate dynamic and geometric models of hand and object is a weak assumption and we show how our method is robust to observation noise and model-inaccuracies.

A. Low-level controller

We implement each motion primitive with a low-level, torque controller. Torque control is inherently more robust to uncertainty by providing compliance. This is empirically evidenced by controllers developed for locomotion and multi-contact control [25, 33, 34].

The low-level controller receives intermediate target object poses X_d or contact configurations C_d from the mid-level

controller. By modelling the dynamics of the hand and object, the controllers can compute the necessary joint torques to achieve X_d or C_d. Following [35], we express the dynamics of the hand when making contact at each fingertip as:

$$M_{hnd}(q)\ddot{q} + V(q,\dot{q}) + J^T\lambda = \tau, \qquad (1)$$

where $M_{hnd}(q)$ is the mass matrix of the hand, V computes the centrifugal and Coriolis terms, $q \in \mathbb{R}^n$ is a vector containing the n joint angles of the hand, J is the stacked Jacobian for each finger tip, λ contains the contact force and moment applied at each contact, and $\tau \in \mathbb{R}^n$ is the torque applied at each joint. Similarly, we express the dynamics of the object as:

$$M_{obj}(X)\dot{\nu}_{obj} - G\lambda = g, \qquad (2)$$

where $M_{obj}(X)$ is the mass matrix of the object, ν_{obj} is the twist of object, G is the grasp matrix, which relates contact forces to a wrench on the object. g contains the external forces applied on the object, including gravity. Then, by plugging Eq. 2 into Eq. 1, we have

$$M_{hnd}(q)\ddot{q} + V(q,\dot{q}) + J^TG^{-1}(M_{obj}(X)\dot{\nu}_{obj} - g) = \tau. \qquad (3)$$

Note that this system has multiple solutions, and has to adhere to additional constraints on the contact force. The normal force λ_N has to be larger than 0, and the shear force has to be smaller than $\mu\lambda_N$ to prevent slipping, where μ is the friction coefficient. To address these constraints, we compute the null space of G, denoted as $\mathcal{N}(G)$, so that for any $\lambda_0 \in \mathcal{N}(G)$, we have $G\lambda_0 = 0$. We choose some λ_0 with a sufficient normal force component such that for $\lambda = \lambda_p + \lambda_0$, where λ_p is a particular solution in Eq. 2, the sheer force component stays within the friction cone, thus avoiding unintended sliding on the contact surface. We use λ as the target contact force per contact point in Eq. 2, which ensures that the hand holds the object firmly during motion and in the presence of some external perturbations.

Reposing: The objective of this manipulation primitive is to change the object pose X while maintaining the contact configuration C. Under this assumption, there is no slip at the fingertips, which means that the twists of the fingertips are the same as of contact points on the objects. This yields $J\dot{q} = \nu_{ftips} = \nu_{contact} = G^T\nu_{obj}$. Considering the definition of the Jacobian $\nu_{ftips} = J\dot{q}$, we differentiate it and obtain $\dot{\nu}_{ftips} = \dot{J}\dot{q} + J\ddot{q}$. Substituting this into (3), we obtain:

$$(M_{hnd}(q)J^{-1} + J^TG^{-1}M_{obj}G^{-T})\dot{\nu}_{ftips} + \\ (V(q,\dot{q}) - M_{hnd}(q)J^{-1}\dot{J}\dot{q}) - J^TG^{-1}g = \tau. \qquad (4)$$

Given a target object state $(X_d, \dot{X}_d, \ddot{X}_d)$, we can compute a desired fingertip acceleration $\dot{\nu}_{ftips}$ using a PID controller: $u_{ftips} = \ddot{X}_d + K_d(\dot{X}_d - \dot{X}) + K_p(X_d - X)$. The error dynamics of this controller are exponentially stable with suitable gain matrices. Substituting u_{ftips} for $\dot{\nu}_{ftips}$ in Eq. 4 gives the appropriate actuation torque.

Sliding: The objective of this primitive action is to change the contact configuration C by sliding one finger on the

Reposing	$+\Delta \boldsymbol{X}^x, -\Delta \boldsymbol{X}^x, +\Delta \boldsymbol{X}^y, -\Delta \boldsymbol{X}^y, +\Delta \boldsymbol{X}^\theta, -\Delta \boldsymbol{X}^\theta$
Sliding	$+\Delta \boldsymbol{C}^1, -\Delta \boldsymbol{C}^1, +\Delta \boldsymbol{C}^2, -\Delta \boldsymbol{C}^2, +\Delta \boldsymbol{C}^3, -\Delta \boldsymbol{C}^3$
Flipping	Flip either left or right finger to the other side

TABLE I: Discretization of manipulation primitives into 14 possible actions.

object surface while keeping the other finger contacts fixed. Given a target contact configuration $\boldsymbol{C}_d$, we compute the desired joint angles $\boldsymbol{q}_d$ using inverse kinematics. We then use inverse dynamic to compute the joint torque

$$\boldsymbol{\tau} = \boldsymbol{M}_{hnd}(\boldsymbol{q})\boldsymbol{u}_{joint} + V(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{J}^T \boldsymbol{G}^{-1}(\boldsymbol{M}_{obj}(\boldsymbol{X})\dot{\boldsymbol{\nu}}_{obj} - \boldsymbol{g}),$$
$$(5)$$

where the control input is $\boldsymbol{u}_{joint} = -\boldsymbol{K}_d \dot{\boldsymbol{q}} + \boldsymbol{K}_p(\boldsymbol{q}_d - \boldsymbol{q})$.

Flipping: The objective of this primitive action is to move one finger off the object surface to a new contact position while using the other fingers to hold the object. This is also referred to as *finger gaiting* [15] or *pivoting* [3]. Compared to the sliding action, this action is more useful when the new contact configuration $\boldsymbol{C}_{t+1}$ is far from the current one $\boldsymbol{C}_t$. Specifically, when manipulating a pole-shaped object, we apply the flipping motion when moving a finger from one side of the pole to the other side. The flipping process can be divided into breaking contact while re-balancing the object and making new contact on the moving object. The flipping primitive can be triggered as a whole by the mid-level controller (as shown in Section IV-C and IV-D) or broken into two motion primitives for more complex objects and triggered separately (as shown in Section IV-E).

B. Mid-level Controller

In our hierarchical structure for in-hand manipulation, we use DRL to learn a policy that can decide the next motion primitive and its parameters, as shown in Fig. 1(a), so that the hand can move the object to the goal pose $\boldsymbol{X}_g$. Given the current state $\boldsymbol{S}$, the policy selects an action $\boldsymbol{A}$ from a discrete set. Each action in that set corresponds to a motion primitive with a specific choice of parameters. Table I lists all the possible actions for moving an object in a plane, i.e. changing its 2D position and orientation. Here, the parameters are either the motion direction of the object or contact or which finger to flip. The motion magnitude Δ is fixed. Given $\boldsymbol{A}$, the lower level uses the aforementioned controllers to reach the provided intermediate target object pose or contact configuration. The agent receives a reward when it reaches the final goal.

Note that some actions will only be successful in specific contact configurations. For example, flipping one finger can only succeed when the two other fingers are opposing each other, so that they can keep the object in relative balance when the flipping finger is released. Therefore, flipping can be successfully attempted much less frequently than reposing and sliding. Nevertheless, the policy has to discover the conditions for a successful flip through trial and error. To address this, we assign a positive reward specifically for a successful and useful flipping action, and a small negative reward for failures. A flipping motion is useful if the new configuration $\boldsymbol{C}$ makes it easier for the object to move towards the goal. Specifically, flipping can be used to give

the hand more freedom to re-orient the object. Thus, if after flipping the normal direction of the flipped finger is aligned with the direction of $\boldsymbol{X}_g^\theta - \boldsymbol{X}_t^\theta$, we consider the flip to be 'useful'. The flipping reward also encourages contact configurations that can make a successful flipping motion when necessary. Another major cause for failure is losing contact, which commonly happens when the target pose is outside the workspace limits. To filter out those invalid primitive choices, we design a feasibility filter that rejects actions at states that are not feasible according to inverse kinematics, which also returns a negative reward to the DRL network. This filter is optional, but useful for pruning actions from the policy's output without changing the environment and forcing the policy to pick feasible actions. This approach is similar to training a policy restricted to legal actions, as done in DQN [36] or AlphaGo [37]. It is not straightforward to include such a filter in an end-to-end framework that directly outputs joint torques without a notion of workspace limits.

IV. EXPERIMENTS

In this section, we report experimental results regarding the following questions: (i) How well does our approach perform in-hand manipulation tasks? (ii) How robust is our model in the presence of observation noise and model inaccuracies on, for instance, object dimensions and weight? And (iii) does our approach generalize to objects of different shapes? We compare to an end-to-end and search-based baseline. Our hypothesis is that our approach is especially advantageous for object goal poses that require complex manipulation sequences.

We implemented our approach in PyBullet [38] in which a three-fingered hand has to move an object within a 2D vertical plane from one pose to a goal pose. Downward gravity is present. Therefore, the object has 3DoF (2D position and 1D orientation) and has to be held during manipulation to not drop. Each finger is fully-actuated and has 2DoF. For the first two experiments, we use a pole-shaped object of 50cm length and 2cm width. For the third experiment, we use a cube.

A. Details on Training Mid-Level Policies and Baseline

We train the proposed mid-level policy using *Proximal Policy Optimization* (PPO) [39] with 16 parallel environments. The observation includes joint angles $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$, current object pose $\boldsymbol{X}_t$ and the goal pose $\boldsymbol{X}_g$. The reward function is as follows

$$r_t = \begin{cases} +5 & \text{if the object reaches the goal at time } t, \\ -0.01 & \text{if the action is invalid at time } t, \\ +1 & \text{if flipping is useful and successful.} \end{cases}$$

An episode is considered successful if the object reaches a pose close to the goal, with a translational error less than 1cm and rotational error less than $5.7°$ (0.1 rad). Since the hand should always firmly grasp the object, we consider dropping the object a failure case. This happens when there is no

	Succ. Rate	Drop Rate	Avg. Time (s)
Ours-Easy	95.4%±0.7%	3.8%±0.7%	7.1±0.2
Ours-Mid	94.6%±0.4%	3.7%±1.3%	8.9±0.6
Ours-Hard	79.3%±0.6%	15.7%±1.4%	13.4±0.3
DDPG-Easy	72.5%±13.5%	19.3%±14.2%	1.82±0.1
DDPG-Mid	61.7%±6.6%	26.8%±10.2%	1.97±0.1
DDPG-Hard	11.1%±6.8%	84.0%±10.8%	2.35±0.1
Search-Easy	91.4%	5.2%	8.1
Search-Mid	79.8%	13.6%	8.1
Search-Hard	54.4%	34.8%	8.25

TABLE II: Success rate (with standard deviation) for 500 episodes of manipulating a pole averaged over 3 random seeds. Dropping is a common failure case when the hand loses necessary contact to keep the object grasped. Average time represents the real-world time to complete a successful manipulation sequence. Experiments are conducted in three groups with an increasing level of difficulty.
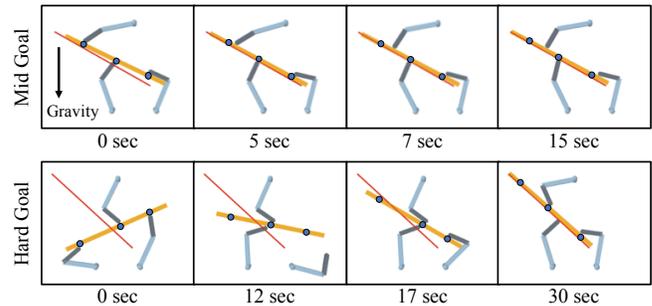

Fig. 3: Snapshots of some example episodes when manipulating the pole to the goal pose. The goal pose is indicated in red and the initial contact points are indicated in blue. The upper episode shows the medium goal and the lower episode shows the hard goal.

contact on one of the sides of the object. It is also a failure if the hand does not reach the goal within a time limit.

Little work towards this problem combines model-based and model-free approaches, so we compare our method with an end-to-end policy trained using *Deep Deterministic Policy Gradient* (DDPG) [40] and a search-based baseline [16], respectively. The DDPG policy operates in the same simulation environment and directly outputs joint torques without encoding any structural priors in the form of traditional controllers or the dynamics of the system. We chose DDPG as it has been shown to work well for continuous control tasks, and for complex dexterous manipulation [14, 28]. The reward function is sparse with $+5$ when reaching the goal otherwise 0. Both policies are represented by 3 fully connected layers with 256 units.

For the search-based baseline, we build a graph by discretizing the feasible object poses $\boldsymbol{X}$ and contact configurations $\boldsymbol{C}$. To trade-off computational complexity and precision, we empirically choose a grid resolution of 2cm and 0.2rad for $\boldsymbol{X}$ and 5cm for $\boldsymbol{C}$. The grid cells form the nodes in the graph. The edges between neighboring nodes correspond to the manipulation primitive that connects them. In-hand manipulation is then equivalent to finding a path from the current pose to the goal pose using Dijkstra's algorithm [41].

B. Dataset

To train our model on reachable goals, we collect a comprehensive dataset of the object's initial poses $\boldsymbol{X}_0$ and goal poses $\boldsymbol{X}_g$. We divide the dataset of $(\boldsymbol{X}_0, \boldsymbol{X}_g)$ pairs into three groups: Easy, Medium, and Hard Goals. In the Easy Goal group, $\boldsymbol{X}_0$ and $\boldsymbol{X}_g$ are in the range of the same contact configuration $\boldsymbol{C}$. Therefore, the goal can be reached by only using reposing. The Medium Goal group contains pairs $\boldsymbol{X}_0$ and $\boldsymbol{X}_g$ that are within the same grasp posture $\mathbb{G}$. Only sliding and reposing may be required to reach the goal pose. The Hard Goal group includes pairs of arbitrary $(\boldsymbol{X}_0, \boldsymbol{X}_g)$ pairs, some of which require flipping to be successful.

C. Reaching Desired Object Poses

We evaluate the performance of our method on the task of moving the object to desired goal poses and compare to an end-to-end and search-based baseline. We report results in Table II. Our method has a consistently higher success and

lower dropping rate. The gap to the baselines is especially high for the hard goals. In fact, the end-to-end method fails almost all cases when flipping is required. The results indicate that our hierarchical method has significant advantages when a complex manipulation sequence is required.

An advantage of the end-to-end method is that the resulting policies yield much faster manipulation sequences as can be seen in the supplementary video. The agent learns to exploit the fast dynamics of the object and frequently pushes it so that it is flying towards the goal pose. However, these dynamic manipulation behaviors will hardly be robust when implemented on a real robotic hand and just as in simulation will likely result in the object being dropped frequently. Since our method uses pre-defined motion primitives for manipulation, it does not learn these fast behaviors. However, it will drop the object less often. In the next section, we also show that the overall approach is more robust as the traditional, compliant torque controllers can cope with some observation noise, external perturbations and modelling errors. We expect this robustness to translate to the real world as well. Another advantage of the end-to-end method is that it can flexibly choose the contact points on the hand and exploit other links of the fingers for manipulation than just the tips. While our method does this to some extend, we aim to explicitly add more manipulation primitives and possible contact points on the finger links.

Also the search-based baseline cannot achieve as high of a success rate and as low of a drop rate compared to our approach. A major disadvantage of the search-based method is that its computational complexity grows in $O(n^6)$ with an increasing resolution of $\boldsymbol{X}$ and $\boldsymbol{C}$. For example, the number of nodes can grow to 10^9 with a 1cm position resolution and 0.1rad angle resolution for the medium goal tasks. However, a lower resolution results in a lower accuracy and therefore in a lower planning performance. For example, the flipping primitive is hard to put in a low-resolution graph because this primitive is conditioned on the two fingers that remain in contact oppose each other.

One limitation of our method is the fixed step size of the manipulation sequence. It may be too small in the early stages of motion, and too large when being already close to the goal. In future work, we will explore continuous midlevel controllers or adaptive step sizes.

Training & Testing	Trained Obj.	Heavier	Lighter	Thicker	Thinner	Longer	Shorter
Without noise $\epsilon_{x,y,\theta}$	94.4%	94.4%	95.8%	91.8%	94.2%	91.8%	95.2%
With noise $\epsilon_{x,y,\theta}$	93.2%	92.4%	94.6%	94.4%	91%	89.2%	92.6%

TABLE III: Success rate for inaccurate object models (geometry and dynamics) and/or observation noise. Experiments are done with the medium goal group. The first row shows results for the model that is trained and tested without observation noise. The second row reports results for the model that is trained and tested with observation noise.

D. Robustness Analysis

In this section, we evaluate the robustness of our method in the presence of observation noise and inaccurate models of the object's geometry and inertia properties. Different from end-to-end models (which typically train one policy per object), traditional controllers require some prior knowledge on the robot hand and object. In the real world, object pose estimation may be noisy and the object and hand models may be inaccurate. A successful manipulation method should be robust to these real-world effects.

We evaluate the robustness of our method by adding observation noise to the object pose X_t and by changing the geometry and inertia parameters of the object from the training conditions. We designed the dynamic controller and trained the DRL mid-level controller as described in Section IV-C (Trained Obj.), and test it on poles that are heavier or lighter (1.1x/0.9x mass), thicker or thinner (1.5x/0.5x width), and longer or shorter (1.1x/0.9x length). In the second experiment, we add uniformly distributed observation noise to the object pose X_t with error bounds of $[-0.5, 0.5]$ (cm) in translation and $[-2.86, 2.86]$ (deg) in rotation. In Table III, we report the success rate of 500 evaluation trials on each of these perturbations. The result shows that our method is very robust to the inaccuracy of the object model as the performance is almost the same. When there is observation noise in the system, the success rate drops a few percentage points but the system still performs well. Observation noise leads to error in the computed contact configurations and thereby adds noise to the low-level dynamic controllers. However, two factors make our approach robust to this. First, the mid-level controller takes state feedback into account at every step and is therefore able to react to unexpected changes. Second, torque control is inherently robust to uncertainty due to compliance. For some cases when one finger makes pre-mature contact, the torque will be achieved earlier than expected and the finger will compliantly give in.

E. Manipulating a cube

Lastly, we test whether the overall approach of combining learned policies with the three available manipulation primitives generalizes to objects that are not pole-shaped. We consider a cube (Fig. 4) as an example. Compared to poles, the cube is shorter (side length of 10cm) and has 4 edges instead of 2, and therefore has more grasp postures $\mathbb{G}$. Ideally, the mid-level controller should learn how to choose the motion primitives and parameters, especially for flipping. But for the sake of faster training for this proof of concept with a small object, we disable sliding since it has a limited benefit on expanding the range of object poses. We also decompose flipping into two stages: releasing
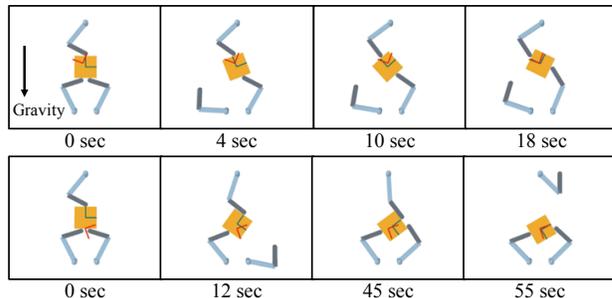

Fig. 4: Examples for manipulating a small cube. The goal pose is indicated in red.

and landing. In the experiment, we choose one initial object pose X_0, and sample any reachable pose X in the range as the goal pose. On a test set of 500 episodes, our method achieves a success rate of 71.4%, and a dropping rate of 20.8%. The average running time is 7.8s. The results show that our overall approach can be generalized to objects of varying shapes. When inspecting the manipulation behaviour (see supplementary video), we find that releasing is often used because it is easy to balance a short object with two fingers. The mid-level controller learned to use the two-fingered primitives to manipulate the cube in most of the cases, because those two finger configurations cover a larger range of object poses X, and they can balance the small cube well. Actually, our method can be extended to other shapes as long as the grasped object is in force closure.

V. Conclusion

We proposed a hierarchical approach to robotic in-hand manipulation, and evaluated our method in simulation with a 3-fingered hand that moves a pole-shaped object in a plane. For the low-level control, our method uses model-based, torque controllers that require some prior knowledge on geometry and inertia of hand and object. They control a set of manipulation primitives. We proposed to learn a mid-level controller with DRL to generate a sequence of these primitives for reaching a distant object goal pose. Using this hierarchical architecture, that it significantly outperformed an end-to-end and search-based baseline while also dropping the object less often. In particular, our approach is better for more distant object goal poses that require complex manipulation sequences. We also showed that our method is robust to observation noise and inaccuracies of the object models in geometry and inertia properties. We also showed how the overall approach can be generalized to objects with different shapes. As future work, we aim to extend the number of manipulation primitives and possible contact points. We also aim to test the approach on a dexterous hand, first in simulation and then in the real world.

REFERENCES

[1] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 255–262.

[2] A. Bicchi, "Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, Dec 2000.

[3] R. R. Ma and A. M. Dollar, "On dexterity and dexterous manipulation," in *2011 15th International Conference on Advanced Robotics (ICAR)*. IEEE, 2011, pp. 1–7.

[4] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers, "Design of the utah/mit dextrous hand," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1986, pp. 1520–1532.

[5] H. Kawasaki, T. Komatsu, and K. Uchiyama, "Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand ii," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 3, pp. 296–303, 2002.

[6] D.-H. Lee, J.-H. Park, S.-W. Park, M.-H. Baeg, and J.-H. Bae, "Kitech-hand: A highly dexterous and modularized robotic hand," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 876–887, 2016.

[7] D. Rus, "Dexterous rotations of polyhedra," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE, 1992, pp. 2758–2763.

[8] Q. Li, C. Elbrechter, R. Haschke, and H. Ritter, "Integrating vision, haptics and proprioception into a feedback controller for in-hand manipulation of unknown objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2466–2471.

[9] D. Rus, "Coordinated manipulation of objects in a plane," *Algorithmica*, vol. 19, no. 1-2, pp. 129–147, 1997.

[10] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: In-grasp manipulation using purely kinematic trajectory optimization," *planning*, vol. 6, p. 7, 2017.

[11] B. Calli and A. M. Dollar, "Vision-based precision manipulation with underactuated hands: Simple and effective solutions for dexterity," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1012–1018.

[12] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 378–383.

[13] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[14] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[15] B. Sundaralingam and T. Hermans, "Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 231–238.

[16] S. Cruciani, K. Hang, C. Smith, and D. Kragic, "Dual-arm in-hand manipulation and regrasping using dexterous manipulation graphs," *arXiv preprint arXiv:1904.11382*, 2019.

[17] T. Okada, "Computer control of multijointed finger system for precise object-handling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, no. 3, pp. 289–299, 1982.

[18] J. Kerr and B. Roth, "Analysis of multifingered hands," *The International Journal of Robotics Research*, vol. 4, no. 4, pp. 3–17, 1986.

[19] S. Leveroni and K. Salisbury, "Reorienting objects with a robot hand using grasp gaits," in *Robotics Research*. Springer, 1996, pp. 39–51.

[20] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 2012, pp. 137–144.

[21] Y. Fan, T. Tang, H.-C. Lin, Y. Zhao, and M. Tomizuka, "Real-time robust finger gaits planning under object shape and dynamics uncertainties," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1267–1273.

[22] L. U. Odhner and A. M. Dollar, "Stable, open-loop precision manipulation with underactuated hands," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1347–1360, 2015.

[23] B. Calli and A. M. Dollar, "Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection," *IEEE Transactions on Automation Science and Engineering*, no. 99, pp. 1–14, 2018.

[24] R. Platt, A. H. Fagg, and R. A. Grupen, "Manipulation gaits: Sequences of grasp control tasks," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 801–806.

[25] A. Simpkins and E. Todorov, "Complex object manipulation with hierarchical optimal control," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2011, pp. 338–345.

[26] M. Murooka, R. Ueda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Global planning of whole-body manipulation by humanoid robot based on tran-

sition graph of object motion and contact switching," *Advanced Robotics*, vol. 31, no. 6, pp. 322–340, 2017.

[27] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 121–127.

[28] A. Rajeswaran*, V. Kumar*, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[29] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[30] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[31] J. K. Salisbury and B. Roth, "Kinematic and force analysis of articulated mechanical hands," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 1, pp. 35–41, 1983.

[32] C. Garcia Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, April 2017.

[33] C. Ott, M. A. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 26–33.

[34] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[35] D. Prattichizzo and J. C. Trinkle, *Grasping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[37] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[38] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[40] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[41] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.