In-Hand Object Rotation via Rapid Motor Adaptation

Anonymous Author(s)
Affiliation
Address
email

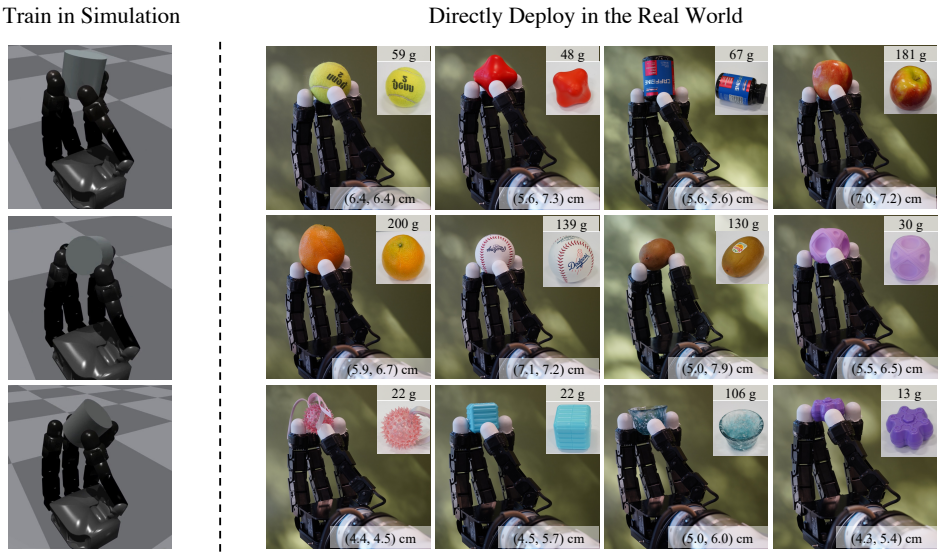


Figure 1: **Left:** Our controller is trained only in simulation on simple cylindrical objects of different sizes and weights. **Right:** Without any fine-tuning, the controller can be deployed to a real robot on a diverse set of objects with convex and nonconvex shapes and different sizes and weights. **Online:** Emergence of natural dynamical finger gaits can be observed in the learned control policy, as shown in the videos at our Project Website.

Abstract: Generalized in-hand manipulation has long been an unsolved challenge of robotics. As a small step towards this grand goal, we demonstrate how to design and learn a simple adaptive controller to achieve dynamic in-hand object rotation using only finger tips. The controller is trained entirely in simulation on only cylindrical objects, which then – without any fine-tuning – can be directly deployed to a real robot hand to rotate dozens of objects with very different object properties, including sizes, shapes, and weights. This is achieved via rapid online estimation and adaptation of the robot's controller to the object properties, using only proprioception history, without any visual or tactile sensing. Furthermore, natural dynamic gaits of fingers automatically emerge from training the control policy via reinforcement learning, without the need for any heuristic rewards or pre-programming for gaits. More results can be viewed at our Project Website.

Keywords: In-Hand Manipulation, Object Rotation, Reinforcement Learning

1 Introduction

Humans are remarkably good at manipulating objects in-hand – they can even adapt to new objects of different shape, size, mass and material with no apparent effort. While several works have shown in-hand object rotation with real-world multi-fingered hands for a single or a few objects [1, 2, 3], truly generalizable in-hand manipulation remains an unsolved challenge of robotics.

In this paper, we demonstrate that it is possible to train an adaptive controller capable of rotating diverse objects with the finger tips of a multi-fingered robot hand (Figure 1). This task is quite

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

challenging for robots since at all times the fingers need to maintain dynamic or static force closure on the object to prevent it from falling (as it can not make use of any other supporting surface such as the palm). Our approach relies on the use of reinforcement learning to train an adaptive control policy in simulation and deploy it on a multi-finger hand for diverse real objects *without* any retraining or fine-tuning via rapid motor adaptation [4]. The core of rapid motor adaptation is the use of an additional extrinsics vector, which is a compressed representation of the object's physical properties, and is jointly learned with the control policy. In [4], this representation is designed for a legged locomotion policy to adapt to different terrains and is continuously updated using the proprioception and action history.

Our key insight is that physical properties such as shape of the objects as perceived by the *finger tips* can be compressed to a low-dimensional space. Despite diversity of real-world objects, for the task of in-hand object rotation, important factors can be defined by a few intrinsic properties of the object. Specifically, we encode the object information including position, size and other physical properties into an extrinsics vector, and train an adaptive policy using reinforcement learning. The learned policy can efficiently rotate different objects in simulation but we do not have access to the extrinsics in the real world. So we learn an adaptation module use the discrepancy between observed proprioception history and the commanded actions to adaptively estimate it. The idea of adaptation using proprioceptive history has been used in several domains [4, 5, 6] but has not yet been explored in the field of in-hand object rotation.

Experimental results on a Wonik's Allegro Hand [7] show that our method is able to successfully rotate over 30 objects of diverse shapes (convex or nonconvex), size (from $4.5\,\mathrm{cm}$ to $7.5\,\mathrm{cm}$), mass (from $10\,\mathrm{g}$ to $200\,\mathrm{g}$) and other physical properties (e.g., deformability) in the real world. We observe a smooth and dynamic finger gaits emerges from the learning process and the learned controller is capable of adapting and generalizing to a wide range of objects. Our approach shows the surprising effectiveness of using only proprioception sensory signals for adaptation to different objects, even without the usage of vision and tactile sensing. To further understand the underlying mechanisms of our approach, we have also studied the estimated extrinsics when manipulating different objects. We show that the estimates can quickly adapts to mass and the diameter of the object, which is critical to generalization as we show in Section 5. We hope that this demonstration of how far a simple adaptive learning can go for dexterous manipulations, even with limited sensory inputs, will motivate further research in the field.

2 Related Work

Classic Control for In-Hand Manipulation. Dexterous In-Hand manipulation has been an active research area for decades [8]. Classic control methods usually need an analytical model of the object and robot geometry to perform motion planning for object manipulation. For example, [9, 10] rely on such a model to plan finger movement to rotate objects. [11] assumes objects are piece-wise smooth and use finger-tracking to rotate objects. [12, 13] demonstrate reorientation different objects in simulation by generating trajectories using optimization. Since the object models needs to be known, many of the above methods are limited in simulation. There have been also attempts to deploy systems in the real-world. For example, [14] calculates precise contact locations to plan a sequence of contact locations for twirling objects using a JPL hand. [15] plans over a set of predefined grasp strategies to achieve object reorientation using two multi-finger hands. [16, 17] use throwing or external forces to perturb the object in the air and re-grasp it. Recently work such as [18, 19] do in-grasp manipulation without breaking the contact. [3] demonstrates complex object reorientation skills using a non-anthropomorphic hand by leveraging the compliance and an accurate pose tracker. The diversity of the objects they can manipulate is still limited due to the intrinsic complexity of the physical world. In contrast to traditional control approaches which may use heuristics or models to solve this task, we instead use model free reinforcement learning to train an adaptive policy to avoid the use of heuristics or simplified models, and use adaptation to achieve generalization.

Reinforcement Learning for In-Hand Manipulation. To get around the need of an accurate object model and physical property measures, in the last few years, there has been a growing interest in using reinforcement learning directly in the real-world for dexterous In-Hand manipulation. [20]

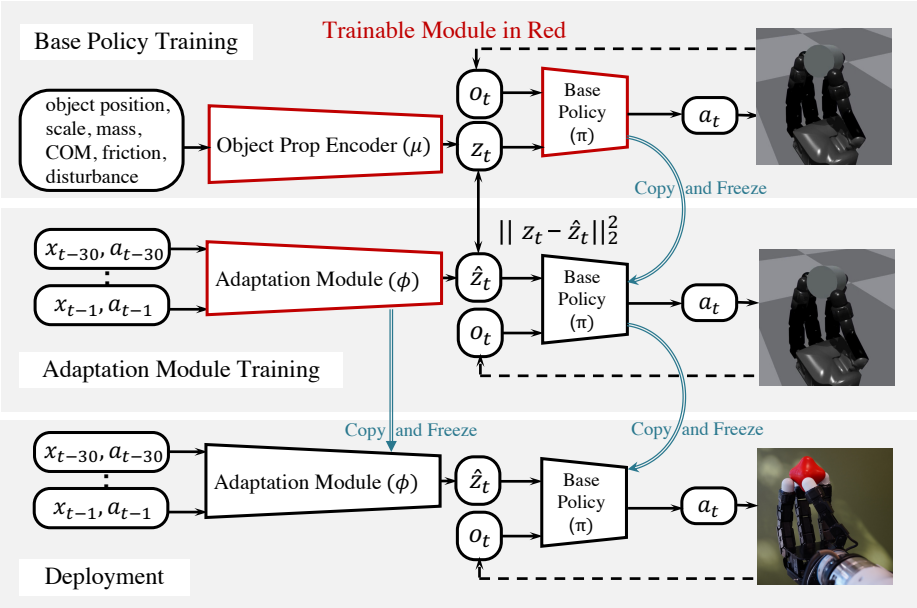


Figure 2: An overview of our approach at different training and deployment phases. In *Base Policy Learning*, we jointly optimize μ and π using PPO [34]. Next, in *Adaptation Module Learning*, with π fixed, we use supervised learning to train ϕ which uses proprioception history to estimate the extrinsics vector $\boldsymbol{z}_t$. During *Deployment*, the base policy π uses the extrinsics $\hat{\boldsymbol{z}}_t$ estimated and updated online by ϕ from proprioceptive history.

learns simple in-grasp rolling for cylindrical objects. [21, 22] learns a dynamics model and plan over it for rotating objects on the palm. [23, 24] use human demonstration to accelerate the learning process. However, since reinforcement learning is very sample inefficient, the learned skills are rather simple or have limited object diversity. Although complex skills such as re-orientating a diverse set of objects [25, 26, 27] and tool use [28, 29] can be obtained in simulation, transferring the results to real-world remains challenging. Instead of directly training a policy in the real world, our approach learns the policy entirely in the simulator and aims to directly transfer to the real world.

Sim-to-Real Transfer via Domain Randomization. There are also plenty of works aim to train reinforcement learning policies using a simulator and directly deploy it in a real-world system. Domain randomization [30] uses a lot of diverse simulation environments and learn reinforcement learning policies that can be deployed in the real-world. Representative examples are [1] and [31]. They leverage massive computational resources and large-scale reinforcement learning methods to learn an agile object reorientation skills and solve Rubik's Cube with a single robot hand. However, they still focus only on manipulating a limited number of objects. [2] learns a finger-gaiting behavior efficiently and transfers to a real robot when hand facing downwards. But they do not only use finger tips and the objects considered are all cubes. Our approach focuses on generalization to a diverse set of objects and can be trained within a few hours.

Sim-to-Real via Adaptation. Instead of relying on domain randomization which is agnostic to current environment parameters, [32] perform system identification via initial calibration, or online adaptive control to estimate the system parameters for Sim-to-Real Transfer. However, learning the exact physical values and alignment between simulation and real-world may be sub-optimal because of the intrinsic inaccuracy of physics simulations. An alternative way is to use a low dimensional embedding which encodes all the environment parameters [5, 33, 4] and learn together with the control policy. This paradigm has enabled robust and adaptive locomotion policies [5, 33, 4] but its potential and advantages in the field of in-hand object rotation has never been explored.

3 Rapid Motor Adaptation for In-Hand Rotation

An overview of our approach is shown in Figure 2. During deployment, our policy infers a low-dimensional embedding of object's property such as position, size, mass from proprioception and

action history, which is then used by our base policy to rotate the object. We first describe how we train a *base policy* with object property provided, then we discuss how to train an *adaptation module* that is capable of inferring these properties. This training scheme shares the same spirit with [4] but we make important and necessary modification for the in-hand object rotation task.

3.1 Learning the Base Policy

Privileged Information. In this paper, privileged information refers to the object's property such as position, size, mass, friction, center of mass, and external forces applied to the object. These information, denoted by a 12-dim vector $\boldsymbol{e}_t \in \mathbb{R}^{12}$ at timestep t, can be accurately measured in the simulation but not in the real world. Instead of using $\boldsymbol{e}_t$ directly, we use an 8-dim embedding (called extrinsics in [4]) $\boldsymbol{z}_t = \mu(\boldsymbol{e}_t)$ which gives better generalization behavior as we shown in Section 5.

Base Policy. Our control policy π takes the current robot joint positions $\boldsymbol{x}_t \in \mathbb{R}^{16}$, the predicted action $\boldsymbol{a}_{t-1} \in \mathbb{R}^{16}$ at the last timestep, together with the extrinsics vector $\boldsymbol{z}_t \in \mathbb{R}^8$ as input, and outputs the relative target changes of the PD Controller (denoted as $\boldsymbol{a}_t$). We also augment the observation to include two additional timesteps to have the velocity and acceleration information. Formally, the base policy output $\boldsymbol{a}_t = \pi(\boldsymbol{o}_t, \boldsymbol{z}_t)$ where $\boldsymbol{o}_t = (\boldsymbol{x}_{t-2:t}, \boldsymbol{a}_{t-3:t-1}) \in \mathbb{R}^{96}$.

Reward Function. We jointly optimize the policy π and the embedding μ using proximal policy optimization [34] with a reward function that depends on several quantities: $\boldsymbol{\omega}$ is the object's angular velocity. $\hat{\boldsymbol{k}}$ is the desired rotation axis (we use the yaw-axis). $\boldsymbol{q}_{\text{init}}$ is the starting robot configuration. $\boldsymbol{\tau}$ is the commanded torques at each timestep. $\boldsymbol{v}$ is the object's linear velocity. Our reward function (subscript t omitted for simplicity) to maximize is then $r \doteq r_{\text{rot}} + \lambda_{\text{pose}}\, r_{\text{pose}} + \lambda_{\text{linvel}}\, r_{\text{linvel}} + \lambda_{\text{work}}\, r_{\text{work}} + \lambda_{\text{torque}}\, r_{\text{torque}}$ where $r_{\text{rot}} \doteq \max(\min(\boldsymbol{\omega} \cdot \hat{\boldsymbol{k}}, r_{\max}), r_{\min})$ is the rotation reward, $r_{\text{pose}} \doteq -\|\boldsymbol{q} - \boldsymbol{q}_{\text{init}}\|_2^2$ is the hand pose penalty, $r_{\text{torque}} \doteq -\|\boldsymbol{\tau}\|_2^2$ is the torque penalty, $r_{\text{work}} \doteq -\boldsymbol{\tau}^T \dot{\boldsymbol{q}}$ is the energy consumption penalty, and $r_{\text{linvel}} \doteq -\|\boldsymbol{v}\|_2^2$ is a object linear velocity penalty.

Note that we do not use any complex reward functions to force the hand to imitate any particular finger gaiting behaviours as in [27]. Yet, a desired stable gaiting behaviour emerges from the energy constraints and penalties on the deviation from the initial pose.

Object Initialization and Dynamics Randomization. We initialize the object and the fingers in a stable precision grasp. Instead of constraining the fingertip positions as in [27], we simply randomly sample the object pose and robot joint position around a canonical grasp until a stable grasp is achieved. During training, we uniformly sample different diameter and side length of the cylinder. We also randomize the mass, center of mass, and friction of these objects (see appendix for the details).

3.2 Learning the Adaptation Module

We cannot directly deploy the so learned policy π to the real world because we do not directly observe the vector $\boldsymbol{e}_t$ and hence we cannot compute the extrinsics $\boldsymbol{z}_t$. Instead, we estimate the extrinsics vector $\hat{\boldsymbol{z}}_t$ from the discrepancy between the proprioception history and the commanded action history via an adaptation module ϕ. This idea is inspired by locomotion literature [4, 5, 35] where the proprioception history is used to estimate the terrain properties. We show that this information can also be used to estimate the object properties. To train this network, we first collect trajectories and privileged information by executing the policy $\boldsymbol{a}_t = \pi(\boldsymbol{o}_t, \hat{\boldsymbol{z}}_t)$ but with the predicted extrinsic vectors $\hat{\boldsymbol{z}}_t$. Meanwhile we also store the ground-truth extrinsic vector $\boldsymbol{z}_t$ and construct a training set $\mathcal{B} = \{\boldsymbol{x}^{(i)}_{t-k:t-1}, \boldsymbol{a}^{(i)}_{t-k:t-1}, \boldsymbol{z}^{(i)}_t, \hat{\boldsymbol{z}}^{(i)}_t\}$. Then we minimize the ℓ_2 distance between $\boldsymbol{z}_t$ and $\hat{\boldsymbol{z}}_t$ using Adam [36]. The process is iterative until the loss converges. We apply the same object initialization and dynamics randomization setting as the above section.

4 Experimental Setup and Implementation Details

Hardware Setup. In our experiments we use an Allegro Hand from Wonik Robotics [7]. The Allegro hand is a low-cost anthropomorphic robot hand with four fingers, with each finger having four degrees of freedom. We use position control for the 16 joints of the robot and convert it to torque using a PD

Method	Within Training Distribution				Out-of-Distribution			
	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)
Expert	$233.71_{\pm 25.24}$	$0.85_{\pm 0.01}$	$0.28_{\pm 0.05}$	$1.24_{\pm 0.19}$	$165.07_{\pm 15.65}$	$0.71_{\pm 0.04}$	$0.42_{\pm 0.06}$	$1.24_{\pm 0.16}$
Periodic	$43.62_{\pm 2.52}$	$0.44_{\pm 0.12}$	$0.72_{\pm 0.21}$	$1.77_{\pm 0.49}$	$22.45_{\pm 0.59}$	$0.34_{\pm 0.08}$	$1.11_{\pm 0.19}$	$1.41_{\pm 0.54}$
NoAdapt	$90.89_{\pm 4.85}$	$0.65_{\pm 0.07}$	$0.44_{\pm 0.11}$	$1.34_{\pm 0.12}$	$54.50_{\pm 3.91}$	$0.51_{\pm 0.06}$	$0.63_{\pm 0.13}$	$1.34_{\pm 0.11}$
DR	$176.12_{\pm 26.47}$	$0.81_{\pm 0.02}$	$0.34_{\pm 0.05}$	$1.42_{\pm 0.06}$	$140.80_{\pm 17.51}$	$0.63_{\pm 0.02}$	$0.64_{\pm 0.06}$	$1.48_{\pm 0.20}$
SysID	$174.42_{\pm 23.31}$	$0.81_{\pm 0.02}$	$0.32_{\pm 0.03}$	$1.29_{\pm 0.72}$	$132.56_{\pm 17.42}$	$0.62_{\pm 0.09}$	$0.50_{\pm 0.09}$	$1.26_{\pm 0.17}$
Ours	$\mathbf{222.27}_{\pm \mathbf{21.20}}$	$\mathbf{0.82}_{\pm \mathbf{0.02}}$	$\mathbf{0.29}_{\pm \mathbf{0.05}}$	$\mathbf{1.20}_{\pm \mathbf{0.19}}$	$\mathbf{160.60}_{\pm \mathbf{10.22}}$	$\mathbf{0.68}_{\pm \mathbf{0.07}}$	$\mathbf{0.47}_{\pm \mathbf{0.07}}$	$\mathbf{1.20}_{\pm \mathbf{0.17}}$

Table 1: We compare our method to several baselines in simulation in two settings: 1) *Within Training Distribution* 2) *Out-of-Distribution*. Our method with online adaptation achieves the best performance compared to all the baselines, closely emulating the performance of the Expert that has the privileged information.

Controller. The stiffness and damping of the PD Controller are 3.0 and 0.1, respectively. The control frequency for the PD Controller is 300 Hz.

Simulation Setup. We use the IsaacGym [37] simulator. During training, we use 16384 parallel environments to collection samples for training the agent. Each environment contains a simulated Allegro Hand and an cylindrical object with different shape and physical properties (the exact parameters are in the supplementary material). The simulation frequency is 120 Hz and the control frequency is 20 Hz. Each episode lasts for 400 control steps (equivalent to 20 s).

Baselines. We compare our method to the following baselines. 1) *A Robust Policy trained with Domain Randomization (DR):* This baseline is trained with the same reward function but without privileged information to be robust to all the shape and physical property variations [1, 2, 31, 38]. 2) *Online Explicit System Identification (SysID):* This baseline predicts the exact system parameters $\boldsymbol{e}_t$ instead of the extrinsic vector $\boldsymbol{z}_t$ during deployment time. 3) *No Online Adaptation (NoAdapt):* During deployment, the extrinsics vector $\hat{\boldsymbol{z}}_t$ is estimated at the first time step and stays frozen during the rest of the run. This is to study the importance of the adaptation module ϕ. 4) *Action Replay (Periodic):* We record a reference trajectory from the expert policy with privileged information and run it blindly. This is to show our policy adapts to different objects and disturbances instead of periodically execute the same action. We also compare with the policy with access to privileged information (*Expert*), as the upper bound of our method (Figure 2, top row).

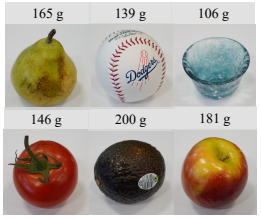
Metrics. We use the following metrics to compare the performance of our method to baselines: 1) *Time-to-Fall (TTF).* The average length of the episode before the object falls out of the hand. This value is normalized by the maximum episode length (20s in simulation experiments and 30s in the real world experiments). 2) *Rotation Reward (RotR).* This is the average rotation reward ($\boldsymbol{\omega} \cdot \hat{\boldsymbol{k}}$) of an episode in simulation. Note that we do not train with this reward. Instead, we use a clipped version of this reward during training. 3) *Radians Rotated within Episode (Rotations).* Since angular velocity is hard to accurately measure in the real world, we instead measure how much radius the policy rotate the object with respect to the world's z-axis. This metric is only used in the real world experiments. 4) *Object's Linear Velocity (ObjVel).* We measure the magnitude of the linear velocity to measure the in-place stability of the object. This is only measured in simulation. The value is scaled by 100. 5) *Torque Penalty (Torque).* We measure the average ℓ_1 norm of commanded torque per timestep during execution to measure the energy efficiency.

5 Results and Analysis

In this section, we compare the performance of our method to several baselines both in simulation and in real-world deployment. We also analyze what the adaptation module learns and how it changes during policy execution and object changes. More qualitative results of our method and several ablations of our method are in our Project Website and the supplementary.

5.1 Generalization via Adaptation

Comparison in Simulation. We first compare our method with the baselines mentioned in Section 4 in simulation. We evaluate all methods under two settings: 1) In the *Within Training Distribution*



Method	Rotations (↑)	TTF (↑)	Torque (↓)
DR	$9.67_{\pm 4.33}$	$0.72_{\pm 0.34}$	$2.03_{\pm 0.36}$
SysID	$10.36_{\pm 2.32}$	$0.61_{\pm 0.33}$	$1.88_{\pm 0.38}$
NoAdapt	N.A.	$0.35_{\pm 0.20}$	N.A.
Ours	$\mathbf{23.96}_{\pm \mathbf{3.16}}$	$\mathbf{0.98}_{\pm \mathbf{0.08}}$	$\mathbf{1.84}_{\pm \mathbf{0.24}}$

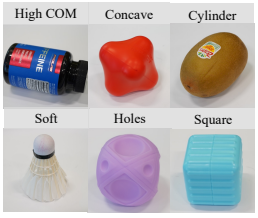
Figure 3: Quantitative Evaluation on a diverse set of Heavy Objects (Left). Our method which uses performs the best in terms of total rotated angle (in radians), Time To Fall (TTF), and Energy efficiency (Torque). The *DR* baseline has a conservative policy which results in a slower angular velocity. *SysID* has a more dynamic and agile policy but very unstable as can be seen in a lower TTF compared to *DR* and ours. The *NoAdapt* baseline fails on the task showing the importance of continuous online adaptation.

setting, we use the same object set and randomization setting as in RL training; 2) In the *Out-of-Distribution* setting, we use objects with smaller and larger shape variation and a larger range of physical randomization range. We also change 20% of the objects to be spheres and cubes. We compute the average performance over 500K episodes with different randomization parameters and initialization conditions. We report the average and standard deviation over five models trained with different seeds.

The results are shown in Table 1. Our method with online adaptation achieves the best performance compared to all the baselines. We see that adaptation to shape and dynamics of the object not only enables a better performance in training, but also gives a much better generalization to out-of-distribution object parameters compared to all the baseline methods. The *Periodic* baseline (i.e., simply playback of the expert policy) does not give a reasonable performance. Although it can rotate the exact same object with the same initial grasp and dynamic parameters, it fails to generalize beyond this very narrow setup. This baseline helps us understand the difficulty of the problem. The *NoAdapt* also has performs poorly compared to our method which uses continuous online adaptation. The weaker performance of this baseline is explained by the fact that it does not update the extrinsics during the episode. This shows the importance of continuously doing adaptation. The *DR* baseline, although roughly matches or performs better than the other baselines in terms of *RotR* and *TTF*, it is worse in other metrics related to object stability and energy efficiency. This is because the *DR* baseline is unaware of the underlying object properties and needs to learn a general policy for all the possible objects. The *SysID* baseline also performs worse than our method in both distributions. This is because it is harder as well as unnecessary to learn the exact values of the shape and dynamics parameters to adapt. This comparison shows the benefit to learn a low-dimensional compact representation which has a coarse relative activation for different physical properties instead of the exact value (see Figure 5 and Figure 6).

Real-World Comparisons. Next, we show real world comparisons of our policy to the baselines on two set of challenging objects shown in Figure 3 and Figure 4. We exclude the *Periodic* baseline because of it does not work well even in Simulation and *Expert* because we cannot access privileged information in the real world. We evaluate each method using 20 different initial grasping positions and 6 different objects. The maximum episode length is $30\,\mathrm{s}$. The results are shown in Figure 3, Figure 4 and the video in our supplementary material.

We first study the behavior of our policy and different baselines in a set of heavy objects (more than $100\,\mathrm{g}$) including a baseball, different fruits, vegetables, and a cup. Our method performs significantly better in all metrics. Our method can achieves consistent rotation for almost all trials without falling down with an average Rotations of 23.96 (with a equivalent rotation velocity $0.8\,\mathrm{rad\,s^{-1}}$). We find the *DR* baseline is very slow and conservative in all the trials since it is unaware of object properties and needs to learn a single gaiting behaviour for all objects. As a result, it has the lowest Rotations, although the TTF is higher than the *SysID* baseline. We also find the *DR* could perform reasonably well on objects of larger sizes because they are easier to rotate and forms a reasonable chunk of training distribution. The *SysID* baseline learns a more dynamic and agile behavior, having a better



Method	Rotations (↑)	TTF (↑)	Torque (↓)
DR	$6.59_{\pm 3.71}$	$0.66_{\pm 0.41}$	$1.85_{\pm 0.37}$
SysID	$8.16_{\pm 3.39}$	$0.46_{\pm 0.36}$	$1.70_{\pm 0.40}$
NoAdapt	N.A.	$0.12_{\pm 0.05}$	N.A.
Ours	$\mathbf{19.22}_{\pm \mathbf{4.08}}$	$\mathbf{0.78}_{\pm \mathbf{0.27}}$	$\mathbf{1.48}_{\pm \mathbf{0.30}}$

Figure 4: Quantitative Evaluation on a diverse set of Irregular Objects (Left). Our method can successfully generalize to rotating a diverse set of objects including non-convex, soft and deformable objects (none of these were included in the training). Our method outperforms the baselines on all the metrics. The *DR* baseline has the second highest TTF but low Rotations because it outputs very conservative and slow trajectories. *SysID* achieves slightly faster but a very unstable policy. The superior performance of our method over baselines shows the importance of adaptation via a low dimensional extrinsics estimation for generalization in this task.

Rotations metric, but a lower TTF, indicating that estimating the exact parameters instead of extrinsics, is hard. It has a similar average angular velocity as our method but a much lower TTF and Rotations showing that precise estimation of system parameters is both unnecessary and difficult. We find that it is particularly hard for it to generalize to the small cup and the slightly soft tomato. Lastly, we observe a similar behavior as in [4] that the *NoAdapt* baseline does not transfer successfully to the real world, showing the importance of continuous online adaptation for success real-world deployment.

We perform the same comparisons on a collection of irregular objects. It contains a container with moving COM, objects with concavity, a cylindrical kiwi fruit, a shuttlecock, a toy with holes, and a cube toy, as shown in Figure 4. Rotating the cube is particularly difficult because we only rely on usage of fingertips. Although these variations are challenging for this task and are beyond what is seen during training, our method still performs reasonably well, outperforming all the baselines. We observe the *DR* baseline can perform stable but slow in-hand rotation for the container and the shuttlecock, but it largely fails for the other objects, indicating its difficulty in shape generalization. For the *SysID* baseline, its stability is significantly lower than both of the *DR* and our method, despite it has a higher angular velocity. For the *NoAdapt* baseline, the behavior is similar as what we observed in the Figure 3.

5.2 Understanding and Analysis

To understand how our method generalizes to a diverse set of objects, we design a few experiments and visualization to help develop some insights.

Extrinsics over Time. We run one continuous evaluation episode in the real world in which we replace the object in the hand every 30s for 6 objects. Note that during training, we never randomize the objects within an episode. During the entire run, we record the estimated extrinsics and plot 2 out of the 8-dim extrinsics vector in Figure 5. The top plot shows the extrinsic value $\boldsymbol{z}_{t,0}$ which responds to changes in object diameter. It has a lower value for smaller diameters and higher value for larger diameters. The bottom plot shows that the extrinsic value $\boldsymbol{z}_{t,2}$ responds to variations in object mass.

Extrinsic Clustering. Another way to understand the estimated extrinsics $\hat{\boldsymbol{z}}$ is by clustering the extrinsics vector estimated while rotating different objects. In Figure 6, we visualize the estimated extrinsics vector using t-SNE for rotating 6 different objects. We find that objects of different sizes and different weights tend to occupy separate regions as explained in Figure 6.

5.3 Real World Qualitative Results

On our Project Website, we qualitatively evaluate our method on 30 objects of different shapes and physical properties, including porous objects and non-rigid objects. The diameter ranges from $4.5\,\mathrm{cm}$ to $7.5\,\mathrm{cm}$ and the mass ranges from $10\,\mathrm{g}$ to $200\,\mathrm{g}$. Note that we only use cylindrical objects of different sizes and aspect ratios during training and show shape generalization in the real world with the use of rapid adaptation. The key insight which allows this generalization is that the shape of the objects as perceived by the fingertips of the hand can be compressed to a low dimensional space. We estimate this low dimensional space from proprioceptive history, which allows us to generalize to objects which are seemingly different in the real world but might look similar in the extrinsics space.

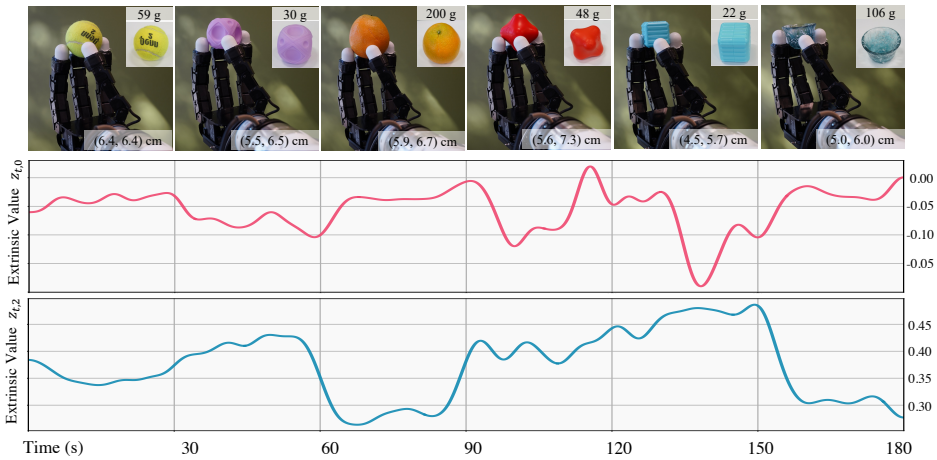


Figure 5: Two of the 8-dim estimated extrinsics vector during one continuous run in which we change the object in the hand every 30s for 6 objects. The top plot shows the extrinsic value $\boldsymbol{z}_{t,0}$ which responds to changes in object diameter with smaller diameters leading to lower values. Since the perceived diameter of the object changes during rotating an irregular shaped object, we see variations in this extrinsic value even within the same object. The bottom plot shows that the correlation between $\boldsymbol{z}_{t,2}$ and object mass: it attains higher values for lighter objects and lower for heavier objects.

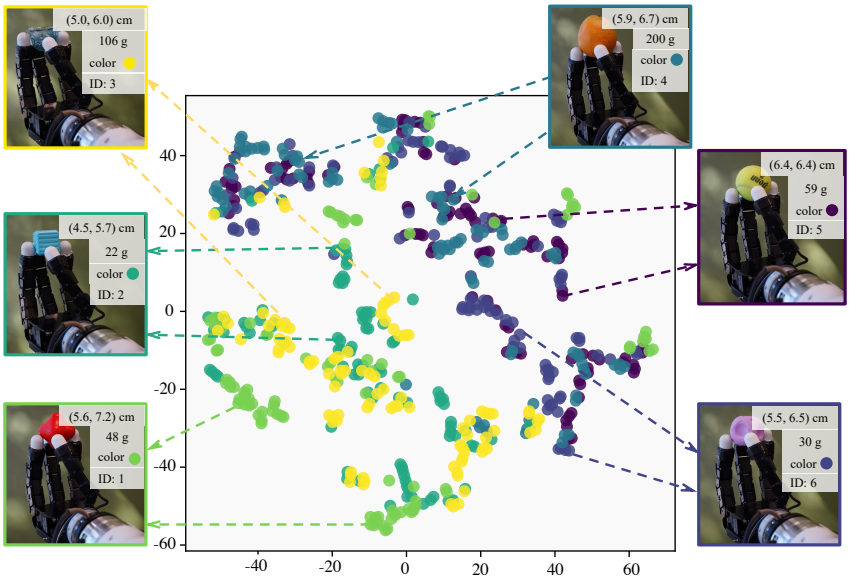


Figure 6: We visualize the estimated extrinsics vector $\hat{\boldsymbol{z}}_t$ using t-SNE for rotating 6 different objects for 5 seconds each. We found that objects of different sizes and different weights tend mostly occupy separate regions. For example, $\hat{\boldsymbol{z}}$ corresponding to smaller objects tend to cluster on the bottom left. The object with ID 1 (shown in the bottom left in the above figure) produces scattered $\hat{\boldsymbol{z}}$ because of its irregular shape, which results in an equivalently varying scale of the object. This locality leads to the scattering of the extrinsics across the t-SNE plot, and this locality is also what gives us generalization as evaluated in Table 1, Figure 3 and Figure 4.

6 Discussion and Limitations

This work shows the feasibility of purely proprioceptive in-hand rotation for a diverse set of objects. Most of our failure cases in the real-world experiments are due to incorrect contact points which leads to unstable force closure. Since our method only relies on the proprioceptive sensing, it is unaware of the exact contact positions between the objects and the finger tips. Another failure case happens when the object is tiny (with diameter $<4.0\,\mathrm{cm}$). In this case the fingers will frequently collide with each other, making the robot not able to maintain the grasp balance of the object. For those more challenging tasks, it might be necessary to incorporate tactile or visual feedback or even both.

References

[1] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning Dexterous In-Hand Manipulation. *The International Journal of Robotics Research (IJRR)*, 2019. doi:10.1177/0278364919887447.

[2] L. Sievers, J. Pitz, and B. Bäuml. Learning Purely Tactile In-Hand Manipulation with a Torque-Controlled Hand. *International Conference on Robotics and Automation (ICRA)*, 2022. doi:10.48550/arXiv.2204.03698.

[3] A. Morgan, K. Hang, B. Wen, K. E. Bekris, and A. Dollar. Complex In-Hand Manipulation via Compliance-Enabled Finger Gaiting and Multi-Modal Planning. *IEEE Robotics and Automation Letters (RA-L)*, 2022. doi:10.1109/LRA.2022.3145961.

[4] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Robotics: Science and Systems (RSS)*, 2021. doi:10.15607/RSS.2021.XVII.011.

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. *Science robotics*, 2020. doi:10.1126/scirobotics.abc5986.

[6] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. Coupling Vision and Proprioception for Navigation of Legged Robots. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] WonikRobotics. AllegroHand. `https://www.wonikrobotics.com/`, 2013.

[8] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An Overview of Dexterous Manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2000. doi:10.1109/ROBOT.2000.844067.

[9] L. Han and J. C. Trinkle. Dextrous Manipulation by Rolling and Finger Gaiting. In *International Conference on Robotics and Automation (ICRA)*, 1998. doi:10.1109/ROBOT.1998.677060.

[10] J.-P. Saut, A. Sahbani, S. El-Khoury, and V. Perdereau. Dexterous Manipulation Planning Using Probabilistic Roadmaps in Continuous Grasp Subspaces. In *International Conference on Intelligent Robots and Systems (IROS)*, 2007. doi:10.1109/IROS.2007.4399090.

[11] D. Rus. In-Hand Dexterous Manipulation of Piecewise-Smooth 3-D Objects. *The International Journal of Robotics Research (IJRR)*, 1999. doi:10.1177/02783649922066268.

[12] Y. Bai and C. K. Liu. Dexterous Manipulation Using Both Palm and Fingers. In *International Conference on Robotics and Automation (ICRA)*, 2014. doi:10.1109/ICRA.2014.6907059.

[13] I. Mordatch, Z. Popović, and E. Todorov. Contact-Invariant Optimization for Hand Manipulation. In *Eurographics*, 2012. doi:10.2312/SCA/SCA12/137-144.

[14] R. Fearing. Implementing a Force Strategy for Object Re-orientation. In *International Conference on Robotics and Automation (ICRA)*, 1986. doi:10.1109/ROBOT.1986.1087655.

[15] R. Platt, A. H. Fagg, and R. A. Grupen. Manipulation Gaits: Sequences of Grasp Control Tasks. In *International Conference on Robotics and Automation (ICRA)*, 2004. doi:10.1109/ROBOT.2004.1307247.

[16] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa. Dynamic Regrasping Using a High-speed Multifingered Hand and a High-speed Vision System. In *International Conference on Robotics and Automation (ICRA)*, 2006. doi:10.1109/ROBOT.2006.1641181.

[17] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge. Extrinsic Dexterity: In-Hand Manipulation with External Forces. In *International Conference on Robotics and Automation (ICRA)*, 2014. doi:10.1109/ICRA.2014.6907062.

[18] C. Teeple, B. Aktas, M. C.-S. Yuen, G. Kim, R. D. Howe, and R. Wood. Controlling Palm-Object Interactions Via Friction for Enhanced In-Hand Manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2022. doi:10.1109/LRA.2022.3143578.

[19] B. Sundaralingam and T. Hermans. Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *Autonomous Robots*, 2019. doi:10.1007/s10514-018-9772-z.

[20] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning Robot In-Hand Manipulation with Tactile Features. In *International Conference on Humanoid Robots (Humanoids)*, 2015. doi:10.1109/HUMANOIDS.2015.7363524.

[21] V. Kumar, E. Todorov, and S. Levine. Optimal Control with Learned Local Models: Application to Dexterous Manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2016. doi:10.1109/ICRA.2016.7487156.

[22] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

[23] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstration. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016. doi:10.1109/IROS.2016.7759557.

[24] V. Kumar, A. Gupta, E. Todorov, and S. Levine. Learning Dexterous Manipulation Policies from Experience and Imitation. *arXiv preprint arXiv:1611.05095*, 2016.

[25] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak. Generalization in Dexterous Manipulation via Geometry-Aware Multi-Task Learning. *arXiv:2111.03062*, 2021.

[26] T. Chen, J. Xu, and P. Agrawal. A System for General In-Hand Object Re-Orientation. In *Conference on Robot Learning (CoRL)*, 2022.

[27] G. Khandate, M. Haas-Heger, and M. Ciocarlie. On the Feasibility of Learning Finger-gaiting In-hand Manipulation with Intrinsic Sensing. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[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 *Robotics: Science and Systems (RSS)*, 2018. doi:10.15607/RSS.2018.XIV.049.

[29] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-Only Imitation Learning for Dexterous Manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, 2021. doi:10.1109/IROS51168.2021.9636557.

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *International conference on intelligent robots and systems (IROS)*, 2017. doi:10.1109/IROS.2017.8202133.

[31] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019.

[32] Y. Karayiannidis, C. Smith, D. Kragic, et al. Adaptive Control for Pivoting with Visual and Tactile Feedback. In *International Conference on Robotics and Automation (ICRA)*, 2016. doi:10.1109/ICRA.2016.7487159.

[33] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 2022.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi:doi.org/10.48550/arXiv.1707.06347.

[35] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik. Adapting rapid motor adaptation for bipedal robots. *arXiv preprint arXiv:2205.15299*, 2022.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

[37] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *arXiv:2108.10470*, 2021.

[38] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg. Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World Trifinger. *arXiv preprint arXiv:2108.09779*, 2021. doi:10.48550/arXiv.2108.09779.