A Additional Results and Analysis

A.1 Per-Object Result Result Analysis.

In Figure 7, we show an almost linear correlation between object mass and the average torque commanded by our policy. This indicates our policy behave differently for objects with different weights and commands smaller torque when object is lighter for energy efficiency.

In Section 5.3, we show the average performance over multiple challenging objects. To give more insights about the working and failure cases of our approach, we analyze the performance on each of the 33 objects we used in experiments. The results are shown in Figure 8, Figure 9, and Figure 10.

Our policy can achieve almost perfect stable and continuous in-hand rotation in 22 out of 33 objects. This set includes objects with different scale, mass, coefficient of frictions, and shapes. Notably, for objects with very high center of mass (the plastic bottle and paper towel), the policy can still perform stable and dynamic in-hand rotations. For more challenging objects including cubes, heavy object with larger aspect ratio (Pear and Kiwi Fruit), and small objects, our policy can still perform about 10 to 20 seconds in-hand rotations. The failure cases are mostly object falling from the fingertips because of incorrect contact positions.

A.2 Multi-Axis In-Hand Object Rotation

We explore the possibility of using our approach to do multi-axis in-hand object rotation. We qualitatively demonstrate preliminary results in the project website. We find this is a harder task than single-axis training and needs about $1.5\times$ training time. Our policy achieves a smooth gait transition between rotation over different axes.

B Ablation Experiments

We analyze different design choices for our method in simulation.

Training with Different Randomization Parameters. Our approach applies a large range of physical randomization during training. In Table 2, we compare the performance difference with our method but without physical randomization.

The *No Rand* entry in Table 2 use the same reward as our method. However, we do not apply any physical randomization during training. The object scale, mass, coefficient of friction, and other physical properties are fixed. This method performs much worse than our method with physical randomization, especially in the out-of-distribution case.

	Within Training Distribution				Out-of-Distribution			
Method	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)
No Rand	171.96$\pm$20.63	0.63$\pm$0.07	0.48$\pm$0.07	1.94$\pm$0.32	88.02$\pm$14.27	0.37$\pm$0.06	0.98$\pm$0.16	2.26$\pm$0.29
Ours	**222.27$\pm$21.20**	**0.82$\pm$0.02**	**0.29$\pm$0.05**	**1.20$\pm$0.19**	**160.60$\pm$10.22**	**0.68$\pm$0.07**	**0.47$\pm$0.07**	**1.20$\pm$0.17**

Table 2: We report the performance of our method without physical randomization during training (the *No Rand* entry). It performs much worse than our method, especially in the out-of-distribution case.

Length of Proprioceptive History. In Table 3, we compare and report the performance with different proprioceptive history length during training the adaptation module. We report the performance a history length 10, 20, and 30. We observe the performance keeps increasing when using a longer history, but saturates at about 30. Considering the efficiency during inference time, we decide to use T=30 in final experiments.

	Within Training Distribution				Out-of-Distribution			
Method	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)	RotR (↑)	TTF (↑)	ObjVel (↓)	Torque (↓)
Ours-T10	215.81$\pm$17.55	0.80$\pm$0.02	0.30$\pm$0.04	**1.19$\pm$0.16**	150.58$\pm$7.07	0.61$\pm$0.05	0.51$\pm$0.08	**1.18$\pm$0.14**
Ours-T20	220.46$\pm$19.72	0.82$\pm$0.02	0.30$\pm$0.04	1.21$\pm$0.17	157.22$\pm$8.11	0.64$\pm$0.04	0.48$\pm$0.07	1.20$\pm$0.15
Ours-T30	**222.27$\pm$21.20**	**0.82$\pm$0.02**	**0.29$\pm$0.05**	1.20$\pm$0.19	**160.60$\pm$10.22**	**0.68$\pm$0.07**	**0.47$\pm$0.07**	1.20$\pm$0.17

Table 3: We compare the effect of using different proprioceptive history length during training the adaptation module. Our approach is not sensitive to this hyperparameter. The performance increases when we increase the history length but saturates when the history length is long enough at T=30.

	Within Training Distribution				Out-of-Distribution			
Method	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$0.16
DR-MLP-T3	176.12$\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$0.20
DR-MLP-T5	165.26$\pm$30.66	0.76$\pm$0.04	0.37$\pm$0.05	1.26$\pm$0.09	115.13$\pm$16.59	0.57$\pm$0.06	0.59$\pm$0.08	1.27$\pm$0.08
DR-MLP-T10	140.95$\pm$13.66	0.72$\pm$0.06	0.39$\pm$0.09	1.52$\pm$0.41	98.32$\pm$10.79	0.54$\pm$0.05	0.57$\pm$0.06	1.51$\pm$0.39
DR-MLP-T20	42.86$\pm$3.66	0.20$\pm$0.02	0.62$\pm$0.03	1.84$\pm$0.19	60.21$\pm$5.19	0.34$\pm$0.05	0.82$\pm$0.09	1.95$\pm$0.20
DR-MLP-T30	49.83$\pm$32.35	0.31$\pm$0.19	1.32$\pm$1.04	1.52$\pm$0.41	36.33$\pm$20.21	0.24$\pm$0.14	1.75$\pm$1.28	2.03$\pm$0.40
DR-LSTM-T10	115.28$\pm$32.99	0.56$\pm$0.15	0.49$\pm$0.08	2.00$\pm$0.45	73.60$\pm$23.88	0.38$\pm$0.12	0.77$\pm$0.19	1.91$\pm$0.41
DR-LSTM-T20	94.33$\pm$33.78	0.44$\pm$0.10	0.57$\pm$0.08	1.93$\pm$0.28	60.18$\pm$17.85	0.28$\pm$0.06	0.91$\pm$0.15	1.85$\pm$0.27
DR-LSTM-T30	75.61$\pm$9.30	0.36$\pm$0.04	0.62$\pm$0.12	1.91$\pm$0.25	48.32$\pm$4.81	0.24$\pm$0.03	1.04$\pm$0.18	1.84$\pm$0.22
Ours	**222.27**$\pm$21.20	**0.82**$\pm$0.02	**0.29**$\pm$0.05	**1.20**$\pm$0.19	**160.60**$\pm$10.22	**0.68**$\pm$0.07	**0.47**$\pm$0.07	**1.20**$\pm$0.17

Table 4: We compare the domain randomization baseline with extended temporal input. However, we find that both the MLP and LSTM networks suffer optimization difficulty. The performance decreases when the input contains longer observation sequences.

Additional Baseline for Robust Domain Randomization (DR). In the main text, the Robust Domain Randomization (DR) baseline only receives $\boldsymbol{o}_t$ as the input, which contains the robot joint positions and actions in the past three timesteps. In Table 4, we study the effect of including longer robot observations.

To fuse robot observations from multiple steps, we consider two different architectures. In MLP, we flatten and concatenate the observations over the time dimension and directly feed it into the policy network. We also explore to use an LSTM network to capture the temporal correlation of input observations. The input will first pass through an LSTM network and then feed into the policy network. The results are shown in Table 4.

We find that the MLP network is hard to optimize via PPO when temporal length is higher. We see a clear trend of decreasing performance when we increase the temporal length from 3 to 30. LSTM performs better than MLP when the temporal length is larger but still suffer the optimization difficulty.

We also consider a temporal convolution network but find it cannot be successfully optimize and only produce random policies.

C Implementation Details

Physical Randomization Parameter. We apply domain randomization during training the base policy as well as the adaptation module. The parameters are listed in Table 5 (Train Range). During simulation evaluation, we use a larger randomization range to test the performance of out-of-distribution generalization.

We use cylindrical objects for training. The cylindrical objects has a radius of $8\,\mathrm{cm}$ and a height sampled from the following discretized list: $[0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.2]\ \mathrm{cm}$. The whole object will be scaled by the randomized object scale parameter specified in Table 5.

Following [1], we apply a random disturbance force to the object during training. The force scale is $2m$ where m is the object mass. The force is decayed by 0.9 every $80\,\mathrm{ms}$ following [1]. The force is re-sampled at each time-step with a probability 0.25. During out-of-distribution testing, we increase the force scale to be $4m$. We also add a noise to the joint position sampled from a uniform distribution $\mathcal{U}(0, 0.005)$ to make it more robust to the real-world noisy joint readings.

We also include 10% sphere objects and 10% cubes in the out-of-distribution evaluation. The canonical sphere object is of diameter $8\,\mathrm{cm}$ and the cube object is of side length $8\,\mathrm{cm}$. They are also scaled by the randomized scale parameter.

Stable Precision Grasp Generation. Our approach assumes the object is grasped at the beginning of the episode. To achieve this, we start from a canonical grasping position using fingers. Then we add a relative offset to the joint positions sampled from $\mathcal{U}(-0.25, 0.25)\mathrm{rad}$. Then we forward the simulation by $0.5\,\mathrm{s}$. We save the grasping pose if all the following conditions are satisfied:

1. The distance between the fingertip and the object should be smaller than $10\,\mathrm{cm}$.

Parameter	Train Range	Test Range
Object Shape	Cylindrical	Cylindrical, Cube, and Sphere
Object Scale	[0.70, 0.86]	[0.66, 0.90]
Mass	[0.01, 0.25] kg	[0.01, 0.30] kg
Center of Mass	[-1.00, 1.00] cm	[-1.25, 1.25] cm
Coefficient of Friction	[0.3, 3.0]	[0.2, 3.5]
External Disturbance	(2, 0.25)	(4, 0.25)
PD Controller Stiffness	[2.9, 3.1]	[2.6, 3.4]
PD Controller Damping	[0.09, 0.11]	[0.08, 0.12]

Table 5: Randomization Range of Physics Parameters. We sample the value of Object Scale, Mass, Center of Mass, Coefficient of Friction, Stiffness, and Damping from a uniform distribution where the lower and upper values are specified in the table. The specification of External Disturbance is specified in the text.

2. At lease two fingers are in contact with the object.
3. The object's height should be above $14.5\,\mathrm{cm}$ higher than the center of the palm.

In practise, we discretized (each region is separated by 0.2) the scales specified in Table 5 and pre-sampled $50,000$ grasping poses for the robot hand and the object for each scale.

Reward Definition. Our reward function (subscript t omitted for simplicity) has the following component:

1. $r_{\text{rot}} \doteq \max(\min(\boldsymbol{\omega} \cdot \hat{\boldsymbol{k}}, r_{\max}), r_{\min})$ is the rotation reward. $\hat{\boldsymbol{k}}$ is a desired rotation axis in the world coordinate. In the experiment, we use $\hat{\boldsymbol{k}} = [0, 0, 1]^\top$. We use $r_{\max} = 0.5$ and $r_{\min} = -0.5$ to clip the rotation reward.
2. $r_{\text{pose}} \doteq -\left\|\boldsymbol{q} - \boldsymbol{q}_{\text{init}}\right\|_2^2$ is the hand pose penalty. $\lambda_{\text{pose}} = -0.3$.
3. $r_{\text{torque}} \doteq -\left\|\boldsymbol{\tau}\right\|_2^2$ is the torque penalty. $\lambda_{\text{torque}} = -0.1$.
4. $r_{\text{work}} \doteq -\boldsymbol{\tau}^\top \dot{\boldsymbol{q}}$ is the energy consumption penalty. $\lambda_{\text{work}} = -2.0$.
5. $r_{\text{linvel}} \doteq -\left\|\boldsymbol{v}\right\|_2^2$ is a object linear velocity penalty. $\lambda_{\text{linvel}} = -0.3$.

Our final reward function is then defined as

$$r = r_{\text{rot}} + \lambda_{\text{pose}}\, r_{\text{pose}} + \lambda_{\text{linvel}}\, r_{\text{linvel}} + \lambda_{\text{work}}\, r_{\text{work}} + \lambda_{\text{torque}}\, r_{\text{torque}}\,.$$

Discussion on Reward Choice. Our reward function contains a few different components. We clip the rotation reward because otherwise the policy will learn to rotate as fast as possible ignoring the stability requirement. Without the pose penalty, the policy performs worse because the learned gait is unnatural and it does not learn to break and establish new contact. Please see an example in this link. The energy penalty and linear velocity penalty greatly decrease the commanded torque and object's linear velocity. This helps encourage the policy to learn a more stable and smooth gait to solve the task, and empirically yields better sim-to-real performance. To speed up training, we terminate the episode when the object falls out of the fingertips (about $14.5\,\mathrm{cm}$ above the palm). Note that this is different when we do evaluation where we reset the episode when the object falls out of fingers (about $10.0\,\mathrm{cm}$) which is more similar to the cases in the real-world.

Network Architecture. The base policy π is a multi-layer perceptron (MLP) which takes in the state $\boldsymbol{o}_t \in \mathbb{R}^{96}$ and the extrinsics vector $\boldsymbol{z}_t \in \mathbb{R}^8$, and outputs a 16-dimensional action vector $\boldsymbol{a}_t$. There are four layers with hidden unit dimension [512, 256, 128, 16]. We use ELU [37] as the activation function. The extrinsics encoder μ is also a three layer MLP with hidden unit dimension [256, 128, 8] and encodes $\boldsymbol{e}_t \in \mathbb{R}^9$ and output $\boldsymbol{z}_t \in \mathbb{R}^8$. We use ReLU as the activation function.

The input to the adaptation module is the robot joint position and actions in past 30 timesteps (corresponding to 1.5 seconds). This module first encode the joint position and action pairs into a 32-dimensional representations for each timestep via a two-layer MLP (with hidden unit dimension [32, 32]). Then, we apply three 1-D convolutional layers over the time dimension to capture temporal correlations in the input. The input channel number, output channel number, kernel size, and stride of each layer are $[32, 32, 9, 2]$, $[32, 32, 5, 1]$, $[32, 32, 5, 1]$. The flattened CNN output is linearly projected

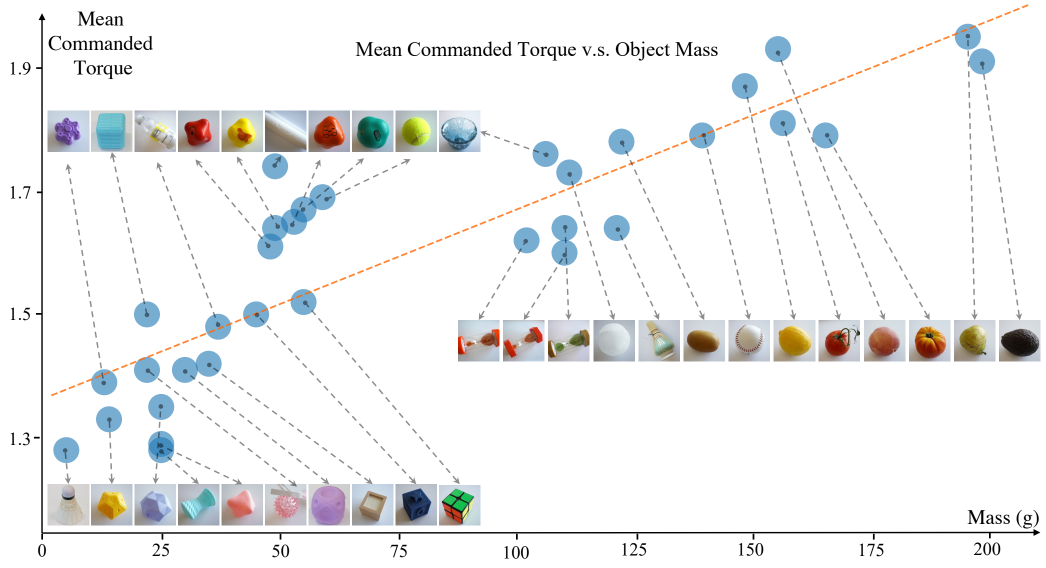

Figure 7: The relationship between object mass and commanded torque. This indicates our policy behave differently for objects with different weights and commands smaller torque when object is lighter for energy efficiency.

by a linear layer to estimate $\hat{z}_t$. We use Adam optimizer [34] to minimize MSE loss. The learning rate is $3e{-}4$.

Optimization Details. During base policy training, We jointly optimize the policy π and the object property encoder network μ using PPO [33]. In each PPO iteration, we collect samples from 16,384 environments with 8 agent steps each (corresponding to 0.4 seconds). We train 5 epochs with a batch size 32,768, leading to 20 gradient updates on this dataset. The learning rate is $5e{-}3$. We optimize for 100,000 gradient updates. It takes about 500 million agent steps which corresponds to roughly 7,000 hours real-world time. The advantage of our approach is that we can utilize the huge amount of simulation samples to learn a complex behavior, which will be infeasible in the real-world.

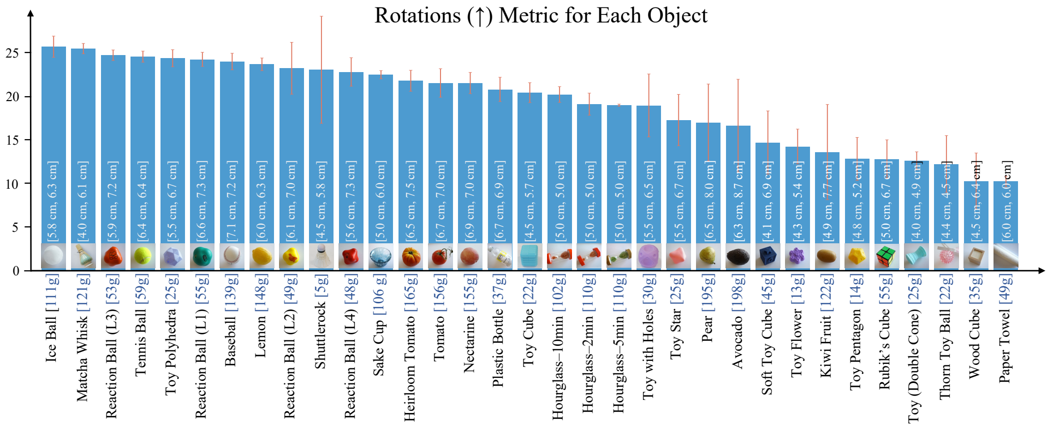

Figure 8: **The performance for each object in terms of the number of rotations achieved**. Each object is annotated with the mass and the shorted and longest diameter of the cross section with the fingertips. We find that sphere and light objects are easier to rotate compared to irregular objects such as avocado, cube, pentagon, and toy flower. The paper towel is with the lowest number of rotation because it keeps colliding with the fingers due to its size (see the video for details).

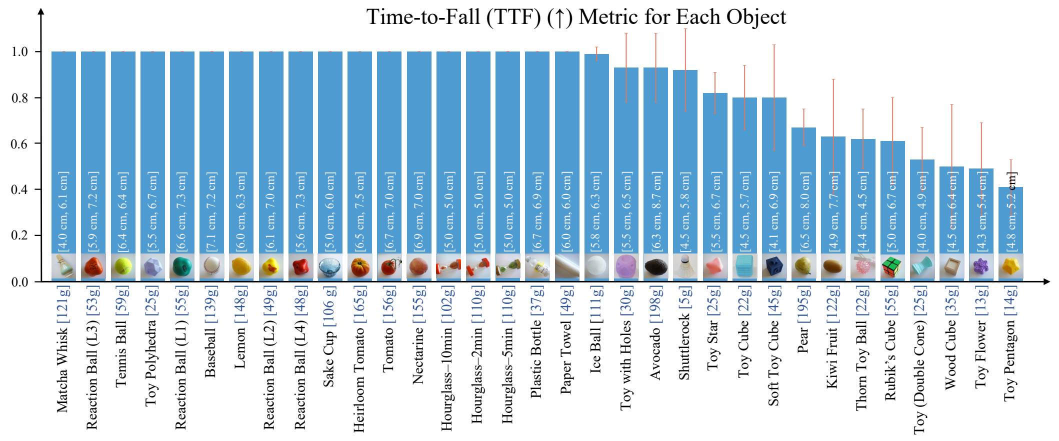

Figure 9: **The stability for each objects in terms of the normalized time-to-fall metric**. Each object is annotated with the mass and the shorted and longest diameter of the cross section with the fingertips. Our policy can keep more than half of the objects stable for more than 30 seconds and rotate more than 15 seconds for all the objects. We also find it's easier to maintain the stability for spherical objects than the cube or pentagon objects.

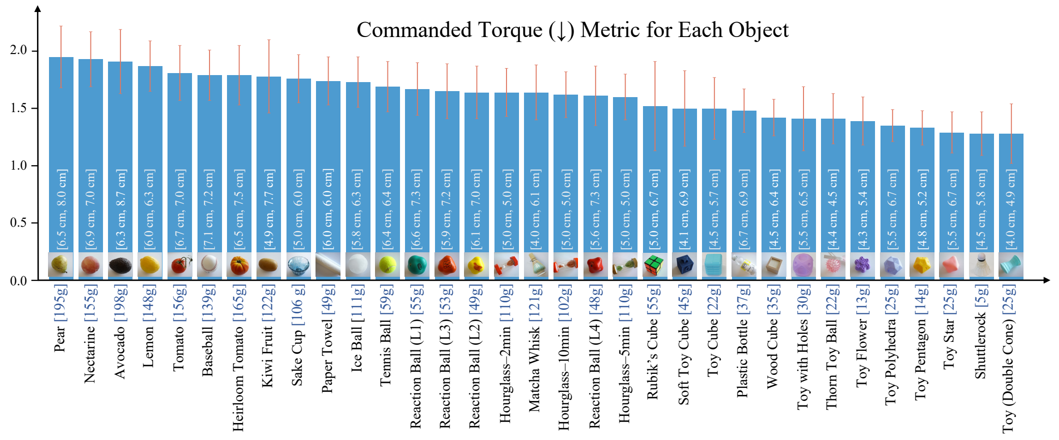

Figure 10: **The commanded torques during rotating different objects**. Each object is annotated with the mass and the shorted and longest diameter of the cross section with the fingertips. We find a general trend that the commanded torques correlate with the mass of the object, which suggests our policy can adapt and command different torques according to the estimated object mass.