Leveraging Tactile Feedback for Robust In-Hand Robotic Manipulation

William Wang and Philippe Wu

***Abstract*—Dexterous in-hand manipulation with high-DOF, non-anthropomorphic hands remains a challenging task in robotics. While prior approaches leverage sim-to-real transfer and domain randomization to train a control policy, they often ignore rich sensor data such as tactile signals available on hardware due to the difficulty of replicating these signals in simulation. In this work, we focus on the task of rotating a lugnut object in a real robotic hand, a useful task for various industrial cases. We propose a two-stage learning framework to incorporate tactile sensing into our manipulation policies. We first train an RL expert policy in simulation and deploy it on a physical robot to collect RGBD, proprioceptive, and tactile data. We then train a behavioral cloning (BC) policy to imitate the RL expert. Through experiments, we show that our BC policy with access to tactile data outperforms the RL expert in manipulation robustness. This approach offers a path for leveraging rich sensor data in manipulation without the need for full sensor simulation.**

I. Introduction

A. Background

Training a robust policy for a highly dexterous robotic hand remains a central challenge in robotics research, particularly for tasks involving intricate manipulation. Notably, OpenAI demonstrated the viability of transferring policies from simulation to reality (sim2real) using domain randomization [4], but their approach did not incorporate many on-hand sensors such as tactile feedback due to the difficulty of replicating these signals in simulation. They instead focused on pose estimation and proprioceptive history. In contrast, we propose leveraging tactile sensors in addition to other sensory modalities in our policy inputs to enhance dexterous manipulation capabilities.

B. Problem Statement

The specific task we target is a lugnut rotation scenario. The task is relatively straightforward: the robot hand must spin a regular hexagonal prism, 40 mm in width, for as long as possible without dropping it. Despite its simplicity, this scenario captures crucial elements of dexterous manipulation, namely, grasp stability, controlled rotation, and adaptive re-gripping. Moreover, a robust policy must be able to resist external disturbances, such as a push on the lugnut.

For manipulation, we use the ROAM Hand 3 (RH3), a 12-DOF, non-anthropomorphic robotic hand platform with custom tactile sensors developed by Robotic Manipulation and Mobility (ROAM) Lab at Columbia. If a policy has access to the on-hand tactile signals, the robot can exploit tactile information to estimate how the lugnut is being grasped and which fingers are in contact. By interpreting tactile feedback, the hand can better coordinate its fingers, allocating some for support (holding up the lugnut) while others execute the rotation.

Our training strategy begins by developing an initial base policy in simulation, using IsaacGym as our platform. Due to the difficulty of controlling a high-DOF, non-anthropomorphic hand, this base policy is trained using RL and called the RL expert. We train the policy using Proximal Policy Optimization (PPO) with domain randomization. Importantly, the RL expert relies only on the joint angles of the robot, the object's z-axis height, and binary contact data. Tactile signals are not used due to the difficulty of simulating the signals accurately in a simulation environment, so a binary contact estimate is instead used as a substitute and estimated based on geometric collision in simulation. On the real robot, this binary contact is estimated from tactile signals. After obtaining a functional policy in simulation, we deploy the RL expert on the real robot and collect the corresponding tactile sensory data that emerge from actual interactions. This creates a dataset that captures how the tactile sensor readings evolve over time when the RL expert controls the manipulations.

With this real-world dataset, we then train a new policy via behavioral cloning that utilizes tactile signals in addition to hand proprioception and RGBD data. By including tactile feedback, we aim to improve robustness and grasp stability. Tactile sensors provide localized information about contact forces and finger–lugnut interactions, which can be invaluable when the robot must maintain its hold on the lugnut while rotating.

To evaluate performance, we define a "grab height," representing the lugnut's initial height above the ground at the start of the trajectory. If this grab height is very high or very low, the hand is barely holding on to the lugnut and must perform active re-gripping or repositioning to achieve a comfortable grasp. We measure how long the robot can manipulate the lugnut without failure under different grab height conditions. By comparing the original RL expert to the new tactile-informed BC policy, we can quantify improvements in terms of sustained rotations, grip robustness, and overall dexterity.

II. Related Work

The task of in-hand manipulation has been studied extensively. As mentioned in the introduction, OpenAI demonstrated in-hand dexterity on real hardware by training a policy via reinforcement learning completely in simulation [4]. Their approach made use of extensive domain randomization to bridge the sim2real gap. Even though the hand they used for manipulation had built-in tactile sensors, they chose to ignore

them completely due to the difficulty of accurately simulating tactile signals.

Another work [3] demonstrated in-hand manipulation using tactile signals on a three-finger "TRX" hand. Their approach involves a high-resolution instrument with dense tactile arrays, which can detect tactile contact on individual patches. They simulate this binary contact by drawing the same array on meshes. However, their approach only works with binary tactile contact, not raw signals.

For our project, we build on work done by the Robotic Manipulation and Mobility (ROAM) Lab at Columbia. The ROAM Hand 3 (RH3) from the ROAM Lab is a custom hardware platform for studying manipulation equipped with tactile, proprioception, and camera data. Similarly to OpenAI, the RH3 makes use of a policy trained via reinforcement learning that allows the hand to twist a lugnut. Both OpenAI and the ROAM lab have yet to make use of rich sensor data embedded in the hardware of the hand due to the difficulties of modeling sensor data in simulation. Our goal is to use the RH3 platform and the existing RL manipulation policy as an expert to train a more robust policy through behavioral cloning that directly makes use of the rich sensor data on the hand.

We draw inspiration for our policy model architecture from BAKU [1]. BAKU is a transformer-based architecture designed for multitask policy learning designed to take in multimodal data for highly efficient training. BAKU encodes multimodal inputs using modality-specific encoders which are then fed through a transformer. The outputs of the transformer goes into the action head which predicts the action at the next time step. BAKU is trained via behavioral cloning where the loss function is formulated as the distance between the sequences of generated actions and the expert policy actions.

III. Hardware and Simulation

A. RH3 Hardware

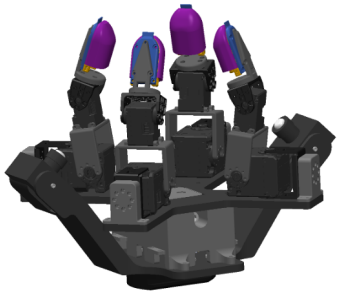

Fig. 1: CAD model of the ROAM Hand 3

The ROAM Hand 3 (RH3) as seen in figure 1 is a non-anthropomorphic robotic hand developed by the Columbia Manipulation and Mobility (ROAM) Lab. The hand features 4 fingers, each with 3 degrees of freedom for a total of 12 degrees of freedom. The fingers are each actuated with two servos, the 2-DOF Dynamixel 2XC-430 and the Dynamixel XC-330, which double as joint encoders for proprioceptive data.

The hand is equipped with 3 cameras. Two fish eye cameras are mounted on the side of the hand offering a side view. A RealSense D405 RGBD camera is mounted in the palm of the hand looking up, providing a measure of the z-axis height of the lugnut on top of RGB images. Custom Spike-a-Tac tactile sensors are equipped at the tip of each finger. The Spike-a-Tacs are capacitor-based tactile sensors that convert pressure into voltage signals. There are seven sensors in each finger for a total of 28 channels on the hand.

B. Simulation

The simulator used to train the expert RL policy is Nvidia's Isaac Gym. Isaac Gym provides actuators with PD controllers but without any trajectory generation. We tune the gains of the physical Dynamixel servo controllers to match the trajectory of the Isaac Gym actuators as much as possible.

IV. Methods

A. Consideration of Approaches

Learning directly from raw tactile signals poses unique challenges. We considered two alternatives before settling on our final approach.

a) Tactile-prediction: Using rollouts from a PPO-trained policy, we fit a lightweight neural network to predict tactile readings from the current observation. Embedding this network in the simulator lets us train a second policy that consumes the predicted tactile feedback. Because the predictor is accurate only near states visited by the original policy, its estimates become unreliable during early exploration, degrading learning.

b) Online real-robot RL: We also explored online RL on the physical hand, by collecting rollouts, updating the policy, and repeating. This option demands extensive manual resets after each failure and is prohibitively time-consuming for our task.

c) Behavior Cloning: We therefore combine simulation-based RL with privileged information and behavior cloning on real-world trajectories as our final approach.

B. Reinforcement Learning Expert Policy

We adopt an asymmetric actor–critic framework with separate multilayer perceptrons (MLPs) for the actor and critic [5]. Policies are optimized with Proximal Policy Optimization (PPO) [6], using Generalized Advantage Estimation (GAE) [7]. Due to the difficulty of simulating raw tactile signals, we substitute tactile data with binary contact estimates. The actor receives binary contact flags, current joint angles and set-points, and object height, and outputs the target joint angles for the next control step.

The agent is rewarded for spinning the object about a target axis while keeping motion, effort, and contact forces low. We clip the angular velocity to avoid runaway incentives, apply a penalty to translational velocity $\mathbf{v}_t$ so the object does not drift as it spins, and use a pose-difference term to improve stability. Effort and work penalties depend on the joint torques $\boldsymbol{\tau}_t$. If

the object falls, we add an out-of-bounds penalty, and a hard-force penalty discourages the hand from slamming the object. At timestep t,

$$r_t = \text{clip}_{v_{\max}}\big(\boldsymbol{\omega}_t \cdot \hat{\mathbf{a}}\big) - \|\mathbf{v}_t\|_1 - \|\boldsymbol{\tau}_t\|_2^2 - d_t - F_t - \mathbb{1}_{\text{OOB}}.$$

Here $\boldsymbol{\omega}_t$ is the object's angular velocity (projected onto the desired axis $\hat{\mathbf{a}}$); all other symbols appear directly in the description above.

Domain randomization is applied to all physical and servo parameters except the object's centre of mass, mass, and scale [8]. PID gains and the velocity/acceleration limits of the Dynamixel servos are tuned to match simulation, further reducing the sim-to-real gap. Hyper-parameters are reported in Appendix table VII.

During hardware roll-outs, object height is estimated in real time by a RealSense D405 depth camera embedded in the palm; a textured test object enhances depth fidelity.

C. Data Collection

We collected lugnut manipulation data on the RH3 using the RL policy, which serves as the expert for twisting the lug nut. We placed the RH3 in a light box to isolate visual disturbances in a controlled and reproducible manner. During executing of the RL policy, we collect proprioceptive, RGBD, and tactile data from the on-hand sensors in real time via ROS2 at 20 Hz. Upon failure of the policy (e.g. dropping the lugnut) we manually reset the lugnut and continue to collect data. We account for this in the data processing phase by slicing out the range of indices consisting of failure cases. Across each trial, we also sought to introduce visual variation within our controlled environment. We change the orientation of the hand in each trial so our policy can learn to be robust against background noise and variation. Figure 2 shows camera data collected from the hand during manipulation trials.

Fig. 2: On-hand data collection running the RL expert policy

After filtering, this preliminary dataset consists of about 18k steps at 20 Hz for 15 minutes of manipulation.

We observed that the main failure cases for lugnut manipulation was the nut falling too low or too high in the hand. The RL expert policy is sometimes able to recover from these states, but unreliably. We curated an additional dataset capturing this recovery behavior to make our policy more robust to common failure modes. We ran the RL policy again while introducing perturbations to the lugnut object by pushing

Fig. 3: On-hand data collection with perturbation

up or down during execution as seen in figure 3. Again, we filtered data only for examples of successful recovery, resulting in 47 examples and and around 26.5k steps at 20 Hz for an additional 22 minutes of data. In total, our combined dataset consisted of 44581 steps, or about 37 minutes of data. We trained our policy on this combined dataset.

D. Tactile Signal Processing

1) Low Dimensional Tactile: The raw tactile signals from the Spike-a-Tacs sensors are extremely noisy. We first calibrate the signals by collecting 2000 steps of raw signals and then grabbing the median. The median is subtracted from all subsequent values. The tactile signals are then normalized using a scaled sigmoid function. We chose an alpha value of 0.016 to support a range of [-137, 137], which was determined by hand by looking at the distribution of tactile values in the dataset.

$$\alpha = 0.016$$

$$\sigma(z) \;=\; \frac{1}{1+\exp(-\alpha\, z)}$$

E. Behavior Cloning Policy Architecture

BAKU introduces a flexible Transformer-based architecture for multi-modal observation [1]. We use the BAKU transformer backbone, but we introduce our own encoders.

1) **Low Dimensional Encoders**: Low dimensional data is composed of the current joint positions and the processed low dimensional tactile signals. A single fully connected layer maps this input to the chosen representation dimension.
2) **Image Encoder**: A ResNet-18 backbone [2] processes visual inputs. The ResNet-18 encoder is not pretrained. The last layer of the encoder is replaced by another fully connected layer that maps the input to the chose representation dimension.

Detailed parameters the policy architecture are given in Appendix table VIII.

F. Training

We train our Behavior Cloning (BC) policy via a standard BC objective, where each demonstration trajectory $\tau = \{(o_t, a_t)\}_{t=1}^{T}$ consists of observations o_t (RGBD, tactile readings, robot joint angles) and expert actions a_t. Here, RGBD

data is drawn completely from the in-palm RealSense camera. We ignore the side fish eye camera observations. We minimize the mean-squared error between the expert trajectory and the model predictions:

$$\text{Let } y_t = [a_{t:t+T_1}],$$
$$\hat{y}_t = \pi_{\text{policy}}(o_{t-T_2:t}) = [a'_{t:t+T_1}].$$

Then the training objective is simply:

$$L_{\text{policy}} = \mathbb{E}_{(o_{t-T_2:t},\, y_t)\,\in\,\mathcal{D}} \left\| \hat{y}_t - y_t \right\|^2$$

Here, T_1 denotes the prediction horizon, T_2 denotes the observation window, a denotes the action. The predictions $\hat{y}_t$ are obtained by passing the observation history $o_{t-T_2:t}$ through $\pi_{\text{T-Tac}}$.

Our policy performs action chunking on the next 10 steps and uses an observation window of 3 with no overlap. Thus, we have $T_1 = 10$ and $T_2 = 3$ During rollout, the policy will store a history of past predicted actions and predict actions using temporal aggregation. Detailed training parameters and setup are given in Appendix table VIII.

We train two versions of the policy, one where o_t consists of [RGBD, tactile readings, robot joint angles] and another where o_t is only [tactile readings, robot joint angles]. We call the first policy BC (with RGBD) and the second policy BC (without RGBD).

G. Evaluation

We reserve 10% of the trajectories for evaluation. For each withheld trajectory, the policy is rolled out step-by-step and its predicted actions are compared with the expert's using mean-squared error (MSE).

V. Experiments

A. Setup

We assess our policies by evaluating their robustness. In our context, we define robustness as the ability to rotate the lugnut object without failure. Failure is defined as a mishandling of the lugnut. Mishandling can be further classified into two types. First is dropping where the lugnut falls. This case mainly occurs when the lugnut drops too low in the hand. Second is tipping, where the fingers lose grip of the lugnut when it rises too high. Here, the lugnut becomes oriented in the incorrect axis and the hand is unable continue turning the lugnut.

To test these policies, we measure how long each policy can manipulate the lugnut before failure occurs. The speed of failure during an episode is highly dependent on the starting position of the object. To systematically test this, we designed a custom adjustable jig as seen in figure 4. The jig ensures that the starting position and orientation of the object is the same among all trials. The jig is also adjustable in the z-coordinate with 10 different height settings (labeled 0-9), allowing us to start the object in abnormally high positions where tipping is likely or abnormally low positions where dropping is likely. These settings allow us to artificially induce a near-failure state at the start of the episode and evaluate how effective a policy is able to recover.

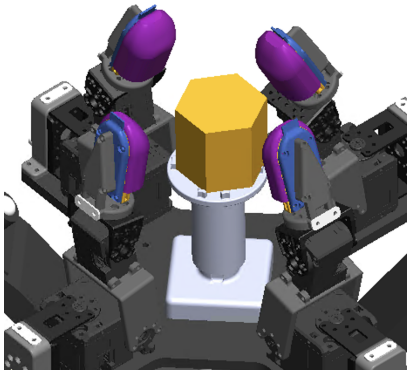

Fig. 4: Custom jig for consistent object starting position with variable z heights

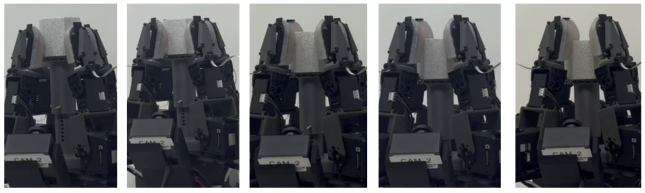

Fig. 5: Five height settings (9, 8, 4, 2, 1) from left to right. 9 and 8 are high near-failure states where tipping is likely, 4 is the normal control height, and 2 and 1 are low near-failure states where dropping is likely.

We conducted manipulation trials using five different height settings with five trials each. Figure 5 shows the five different height settings on the hand. For each trial, we measure the seconds of policy roll out until failure. We cap our trials to 300 seconds where we automatically consider the episode a success.

We perform the experiment on the RL expert, the BC (without RGBD) policy, and the BC (with RGBD) policy. The RL expert policy serves as the baseline to which we compare our BC policies.

B. Results

Our results are plotted in figure 6. The raw data and averages can be found in Appendix tables I, II, III, IV, and V. We note that the averages shown in the figures are a low estimate of the true average. For trials that did not fail, we artificially stopped the episode at 300 seconds due to time constraints. The actual manipulation time before failure could have been much longer, pushing the true average for some of these trials higher.

From our results, we see that our BC (with RGBD) policy consistently outperformed our BC (without RGBD) policy. We also see from the results that the BC (with RGBD) policy performed similarly to the RL expert policy at height settings 4 and 8, but outperformed the RL expert by a significant margin at extreme heights 9, 2, and 1. These results indicate that our

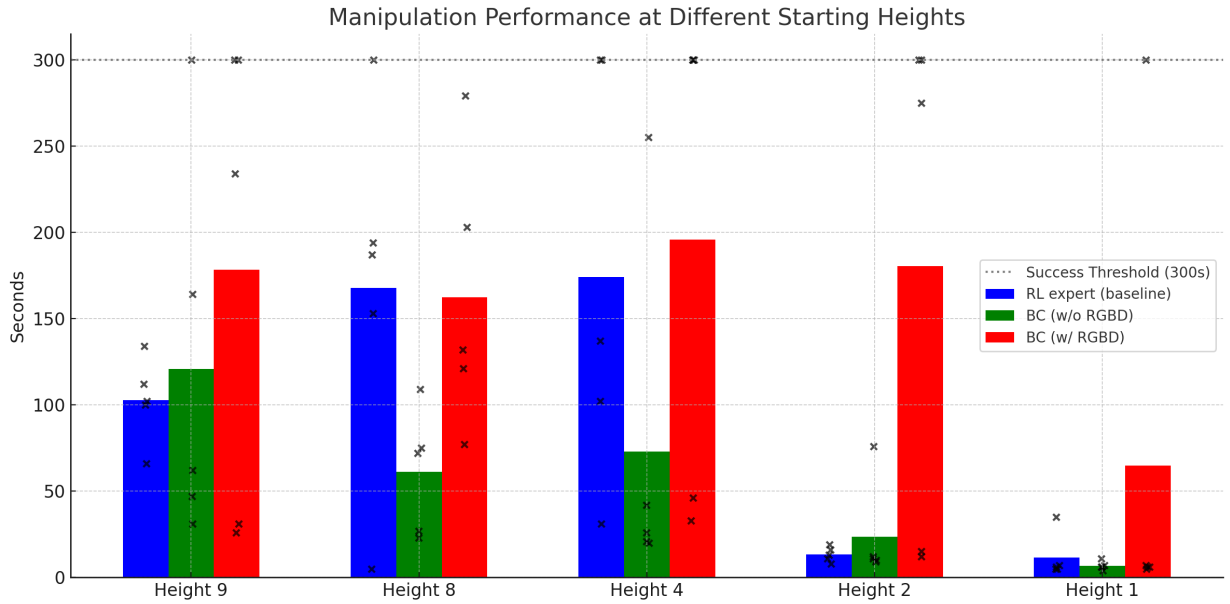

Fig. 6: Experiment results (number of seconds until failure) of all five test heights on each of the three policies are plotted above. The bars show the average time across all five trials. The dotted line at 300 seconds indicate that a trial reached the 300 second threshold and was manually stopped.

BC (with RGBD) policy is more robust than the RL expert as it is better able to recover from near-failure states on both high and low height extremes. [1]

C. Discussion

The success of the BC (with RGBD) policy over the BC (without RGBD) policy indicates that image and depth information is important to the robustness of the manipulation. The RealSense camera gives an accurate estimate of depth of the z-axis height of the lugnut for both the RL and the BC policies. These results show that knowledge of the lugnut height is important for the robustness of the policy. Intuitively, this result makes sense as height is a main determining factor of when the lugnut is close to a failure state.

The improved performance of our BC (with RGBD) policy over the RL expert at extreme height settings suggests that access to rich tactile data allows the BC policy to better recover from near-failure cases when compared to the binary tactile information that the RL expert has access to. We hypothesize that the rich tactile information from sensors allows our BC model to have better knowledge of the current state compared to the RL expert, improving robustness.

We acknowledge a few weaknesses in our experiment. First, as mentioned previously, our calculated averages are a low estimate for the true average manipulation time before failure across our trials. For each height setting, the best performing trial by average seconds also had the greatest number of instances where manipulation did not fail within 300 seconds, so including the true time is unlikely to change the performance rankings. That is, the average manipulation time for the best performing policy across all height settings is a conservative estimate when compared to the other policies within that same experiment. Second, the standard deviation between trials is high, so five trials does not provide confidence that the results accurately represent the true means. Given more time, we propose performing a much larger number of trials to ensure that the differences the policies are more statistically significant.

[1] Video comparison of the RL expert policy and the BC (with RGBD) policy is available here: https://www.youtube.com/watch?v=ldG13yCEtFQ

D. Ablation Studies

To gauge real-world robustness we ablated sensing modalities and measured how long the robot could keep the object aloft (mean $\pm$ s.d. over three trials; initial drop height = 4 cm).

Policy	Sensor ablation	Survival (s)
Baseline RL Expert	Palm Depth covered	7.0 ± 0.6
BC (w/ RGBD)	Palm RGB–D covered	$\mathbf{15.5 \pm 0.8}$
BC (w/o RGBD)	Tactile signals zeroed	11.3 ± 0.5
BC (w/ RGBD)	Normal lighting (no light-box)	203.6 ± 0.4

Covering the palm RGB–D camera sharply degrades vision-centric policies, yet the multisensory *BC (w/ RGBD)* variant still more than doubles baseline performance. Zeroing tactile inputs weakens the *BC (w/o RGBD)* model, confirming its dependence on contact cues. Surprisingly, *BC (w/ RGBD)* shows no loss outside the light box, indicating good generalization to visual disturbances. Overall, robustness is preserved provided every modality the policy expects remains available.

VI. Conclusion

Our work demonstrates a way to train a tactile-based manipulation policy using using behavioral cloning by distilling

an RL expert teacher policy trained with PPO and Domain Randomization. By running rollouts with the teacher policy on real hardware, we can train a student policy that utilizes the rich tactile signals generated. Running experiments, we saw an improvement in robustness when we incorporating tactile signals, allowing our policy to outperform the RL expert teacher. However, we are still limited by the initial performance of the teacher policy. If the teacher policy is unable to produce acceptable or good states initially, our behavior cloning policy would have no dataset to train with. To address this, we are working on incorporating offline Q-learning to do RL finetuning on our policy. Offline Q-learning will allow us to leverage both the bad and good states produced by our policy.

VII. Acknowledgments

We would like to thank Zhanpeng He and Joaquin Palacios for allowing us to use the RH3 platform, for their previous work on the RH3 hardware and RL expert policy, and for their guidance and insight throughout this project.

References

[1] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[3] Wenbin Hu, Bidan Huang, Wang Wei Lee, Sicheng Yang, Yu Zheng, and Zhibin Li. Dexterous in-hand manipulation of slender cylindrical objects through deep reinforcement learning with tactile sensing, 2023. URL https://arxiv.org/abs/2304.05141.

[4] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, 2018. URL http://arxiv.org/abs/1808.00177.

[5] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning, 2017. URL https://arxiv.org/abs/1710.06542.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[7] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

[8] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

APPENDIX

TABLE I: Performance at Height 9

Trial	RL	BC (no RGBD)	BC (w/ RGBD)
1	102	300	300
2	134	164	234
3	100	47	26
4	66	31	300
5	112	62	31
Average	102.8	120.8	178.2

TABLE II: Performance at Height 8

Trial	RL	BC (no RGBD)	BC (w/ RGBD)
1	194	72	132
2	300	75	77
3	187	23	121
4	153	27	279
5	5	109	203
Average	167.8	61.2	162.4

TABLE III: Performance at Height 4

Trial	RL	BC (no RGBD)	BC (w/ RGBD)
1	300	21	46
2	300	26	300
3	137	255	300
4	102	20	33
5	31	42	300
Average	174	72.8	195.8

TABLE IV: Performance at Height 2

Trial	RL	BC (no RGBD)	BC (w/ RGBD)
1	8	76	300
2	16	10	15
3	11	12	300
4	13	9	275
5	19	11	12
Average	13.4	23.6	180.4

TABLE V: Performance at Height 1

Trial	RL	BC (no RGBD)	BC (w/ RGBD)
1	5	4	7
2	35	7	300
3	6	11	5
4	5	6	6
5	7	6	6
Average	11.6	6.8	64.8

Input feature	Critic	Actor
Hand joint position	✓	✓
Hand joint velocity	✓	×
Target joint position	✓	✓
Fingertip contact force	✓	×
Object position	✓	×
Object height (z)	✓	✓
Object orientation	✓	×
Object linear velocity	✓	×
Object angular velocity	✓	×
Target orientation	✓	×
Target orientation Δ	✓	×
Joint torque	✓	×
Fingertip contact (bool)	✓	✓
Fingertip contact position	✓	×
All contacts (bool)	✓	×
Object keypoints	✓	×

TABLE VI: Observation features available to the privileged critic and to the actor network.

Parameter	Meaning	Value
γ	Discount factor	0.99
τ	GAE–λ / target smoothing	0.95
Learning rate	Adam step size	2×10^{-4}
KL threshold	Early-stopping KL limit	0.016
Rollout horizon (T)	Steps collected per update	8
Minibatch size	Samples per gradient step	32 768
Actor epochs (E_π)	Passes over data for policy	5
Critic epochs (E_V)	Passes over data for value	8
Clip coefficient (ϵ)	PPO ratio clip	0.2
Entropy coefficient	Exploration regulariser	0
Gradient-norm clip	$\|g\|_{\max}$	1.0

TABLE VII: Baseline RL expert policy hyper-parameters.

Parameter	Value
Batch size	64
Number of Iterations	10,000
Learning Rate	1e-4
Action horizon	10
History length	3
Representation Dimension	512
Head Dimension	256

TABLE VIII: BAKU parameters