Rotating without Seeing:
Towards In-hand Dexterity through Touch

Zhao-Heng Yin[1,*,†], Binghao Huang[2,*], Yuzhe Qin[2], Qifeng Chen[1], Xiaolong Wang[2]

[1]HKUST   [2]UC San Diego
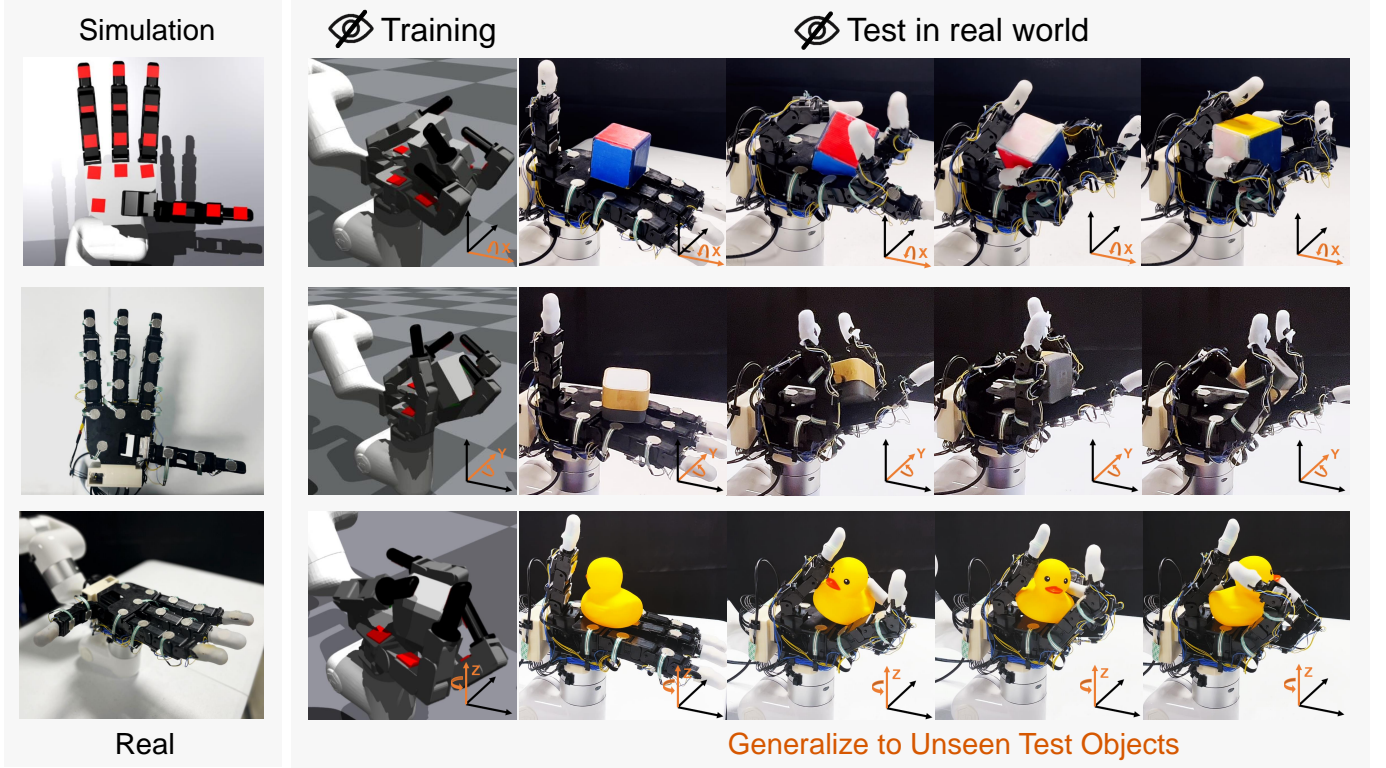
http://touchdexterity.github.io

Fig. 1: We propose **Touch Dexterity**, a new dexterous manipulation system to perform in-hand object rotation with only touch sensing. On the left, we show our hardware setup with 16 FSR sensors attached to an Allegro hand. We train our policy in simulation on rotating diverse objects around different axes. Our trained policy can be directly transferred to the real robot hand and can rotate novel/unseen objects successfully.

Abstract—Tactile information plays a critical role in human dexterity. It reveals useful contact information that may not be inferred directly from vision. In fact, humans can even perform in-hand dexterous manipulation without using vision. Can we enable the same ability for the multi-finger robot hand? In this paper, we present Touch Dexterity, a new system that can perform in-hand object rotation using only touching without seeing the object. Instead of relying on precise tactile sensing in a small region, we introduce a new system design using dense binary force sensors (touch or no touch) overlaying one side of the whole robot hand (palm, finger links, fingertips). Such a design is low-cost, giving a larger coverage of the object, and minimizing the Sim2Real gap at the same time. We train an in-hand rotation policy using Reinforcement Learning on diverse objects in simulation. Relying on touch-only sensing, we can directly deploy the policy in a real robot hand and rotate novel objects that are not presented in training. Extensive ablations are performed on how tactile information help in-hand manipulation.

I. INTRODUCTION

Imagine we are washing the used pan in the kitchen after dinner. Suddenly, the power is cut off unexpectedly, and all the lights go out. What would we do? Most of us may stop the work, put down the pan in the sink, and then probably find our phone in the pocket to light up the way. Simple as it may seem, this sequence of actions actually requires precise execution of in-hand dexterous manipulation in the dark, where we receive no vision input for guidance. Even in normal situations with lights on, the manipulation of objects in hand often comes with heavy occlusions. Without relying on vision, we humans are still very good at feeling and manipulating

objects by hand, which is made possible by the tactile (touch) information coming from our skin. Previous studies in biology also confirm the vital importance of touch information for dexterous manipulation [30]. Can we enable robots with such dexterity with touch sensing?

Indeed, tactile sensing has been a long-standing topic in robotics. With different designs of tactile sensors, robots are able to manipulate objects more precisely using contact information [19, 31, 20] and even complete tasks in a touch-only setup [35, 42]. However, it is still very challenging for touch-only approaches to achieve complex and high degree-of-freedom (DOF) in-hand manipulation. While most current literature focuses on modeling precise and fine-grained contact using increasingly high-quality sensors, it introduces two challenges to in-hand manipulation: (i) Most approaches are only able to attach the expensive sensors to the finger-tips of the gripper or hands instead of covering the whole manipulator, limiting the range of tasks to perform; (ii) It often requires a large number of training samples for complex tasks, but it is hard to leverage a simulator given the Sim2Real gap is usually very large for a delicate sensor.

In this paper, we present **Touch Dexterity**, a new system design and learning pipeline for in-hand rotation using only touching. Instead of using a few sensors on finger-tips that give high-quality patterns [75, 33, 46], we propose the alternative: Use a lot of low-cost binary force sensors (touch or no touch) attached over one side of the hand (fingertips, links, and palm) as shown in Fig. 1 (left). Specifically, we attach the Force-Sensing Resistor (FSR) sensors, which cost around $12 each on Amazon[1], to the Allegro robot hand. Our insight is that, while one single binary force sensor cannot do much, the combination of 16 of them has a strong representation power (2^{16} types of states in maximum), which might allow the robot hand to "feel" the object state without seeing. Importantly, the Sim2Real gap by using such a binary sensor is minimized to the extreme, which allows large-scale sample collection in simulation for training.

With this system setup, we focus on the task of rotating an "unseen" object around the x, y, and z-axis using the multi-finger hand as shown in Fig. 1 (right). Here "unseen" not only indicates there is no vision, but also means the object is not presented during training time. While this task is a simplified version of the in-hand re-orientation task, it is still very challenging as all the fingers are moving with a relatively large motion to rotate the object and prevent it from falling off the palm at the same time. We believe the same pipeline can be directly extended to more complex tasks in the future. We train our policy on multiple objects in parallel in the IsaacGym simulator [38] using Reinforcement Learning (RL), and the learned policy can be directly deployed on the real robot manipulating diverse unseen objects. The key to achieving such generalization across objects and to the real robot is our touch sensors. Our RL policy takes both the binary touch sensing information and the robot's internal state as input and predicts

the action in each time step for closed-loop control. With a large coverage over the object using the touch sensors, our hypothesis is that the policy implicitly learns to understand the 3D structure and pose of the object and perform rotation accordingly.

In our experiments, we test the real-world system with 10 diverse objects. Our method shows surprising robustness in rotating unseen objects using only touch sensing. For example, we can rotate the rubber duck for two cycles without falling, even if it is never presented in training (last row in Fig. 1). We perform extensive ablations on our sensor to validate our design, including disabling all the touch sensors, disabling part of them, and using continuous signals instead of binary signals.

II. RELATED WORK

Dexterous Manipulation Dexterous manipulation has been a long-standing problem in robotics [56, 45, 17, 27, 1, 16, 26, 25, 10, 21, 59, 12, 69, 2]. Among these works, dexterous in-hand manipulation receives a lot of attention in recent years [32, 6, 5, 41, 2]. Several early methods propose to tackle the in-hand manipulation problem with analytical model-based approaches [32, 5]. Nevertheless, they pose certain hypotheses about the objects and the controllers, which makes it hard to scale to more complex tasks. To overcome this limitation, deep Reinforcement Learning has been applied recently on dexterous manipulation [2, 28, 15, 52, 14, 51]. Building on these works, incorporating demonstrations in with imitation learning also leads to better sample efficiency and more natural manipulation behaviors [54, 55, 4, 53, 71, 74, 37, 48, 3]. However, most in-hand manipulation methods are still highly relying on visual inputs [2, 28, 14]. For example, Chen et al. [14] propose to perform in-hand object re-orientation using depth image input, and new hardware is designed to avoid heavy occlusion. Instead of relying on vision which faces the occlusion problem with general hardware, recently, several works [51, 61, 50] propose to perform in-hand object rotation without both visual and explicit tactile sensing. The idea of these works is that we can infer the object's information from the implicit tactile information inside proprioception data. However, these works either only consider object rotation on the fingertip with relatively small finger motion, or the rotation of a limited set of objects. Compared to these works, our system explicitly use touch sensors to percept hand-object interaction, and can solve the object rotation problem on the palm for diverse types of objects, which involves complex object motion and is more challenging. We also find that using explicit tactile sensing enables our touch-based policy to generalize to unseen objects, which is not shown by previous works.

Tactile Robotic Manipulation Biological evidence suggests that tactile information is crucial for the success of human dexterity [30]. This basic observation naturally motivates the research of tactile robotic manipulation [13, 40, 19, 7, 44, 76, 47, 65, 64, 73, 63, 70, 67, 62, 29, 47, 42, 66, 34, 23, 8]. A fundamental question is what kind of touch information is essential. Existing works propose to extract local geometry,

force and torque, contact event, and material properties with various sensors to help manipulation [35]. Different from these works, we find that even using the simplest binary contact signal provided by a sparse sensor array can be helpful for a high-dimensional manipulator. This is also found in [49, 22, 36] where binary contact signals are used for manipulation, object tracking and exploration. However, they focus on low-DOF manipulators rather than a multi-finger robot hand. In the dexterous hand research, Buescher et al. [11] develop a skin-based tactile sensing system on the Shadow hand, which has a similar but denser sensor layout over the palm compared with our work. However, it is still unclear how to use it with a control method to solve in-hand rotation as in our work. Another important question in tactile robotic manipulation is how to simulate the tactile event so as to perform Sim2Real transfer. Researchers have proposed many approaches and strategies for tactile simulation [39, 24, 72, 18, 60, 9]. For example, Xu et al. [72] propose a method to simulate normal and shear tactile force field on the contact surface. Compared with these works, our method does not require any extra simulation design but can leverage the built-in contact simulation of an existing physics simulator.

III. TACTILE DEXTEROUS MANIPULATION SYSTEM

A. Real-word System Setup

Our hardware setup consists of a XArm robot arm and a 16-DOF Allegro Hand with a contact sensor array. The array consists of 16 contact sensors, which are attached to different parts of the allegro hand including the palm and tips as shown in Figure 1(left). The used contact sensors are based on Force-Sensing Resistors (FSR), whose resistance will change when an external force is applied to its surface. These sensors are very sensitive to force and widely used in robotics. We use an STM32F microcontroller to collect the analog voltage signals of each sensor and then forward digital signals to the host.

While these contact sensors are able to output the **continuous** contact force measurement, the signals are usually nonlinear and noisy. As a result, it should undergo necessary preprocessing before being used for control. We **binarize** these measurements with respect to a selected threshold θ_{th} and use this binary contact signal for control. The advantage of using binary signals is that it can reduce the gap between the simulation and the real robot, and simplify the Sim2Real transfer procedure. When using the exact force measurement as observation, it is difficult to align the measurement between the simulation and the real robot, especially since there are still errors in aligning the analog voltage signals to the exact force measurement. In contrast, we can easily calibrate the binarized measurement by adjusting the threshold.

B. Simulation Setup

In this paper, we use the IsaacGym simulator [38] for the training of our tactile manipulation system. The simulation setup is shown in Figure 1(left). We simulate each contact sensor as a fixed link on the finger and palm links. We fetch the net contact force $F = [F_x, F_y, F_z]$ over each sensor link

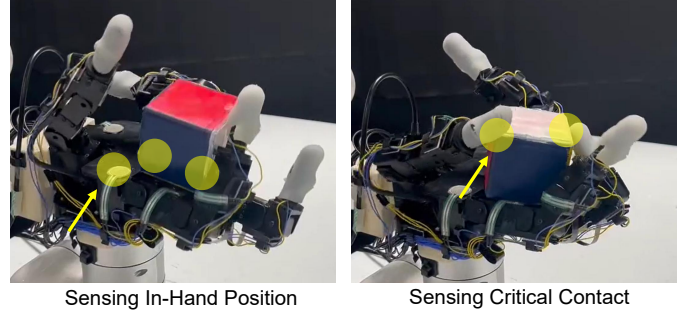Sensing In-Hand Position　　　　Sensing Critical Contact

Fig. 2: Two major functionalities of our sensors: sensing (i) the objects' in-hand position, and (ii) the critical contact during the dexterous manipulation process. Note that we use finger cots to increase the friction and we still have force-sensing resistors inside the finger cots.

provided by the simulator at each simulation step, and use $\|F\|$ as the simulated contact force measurement. Then, we binarize the measurement with another threshold $\tilde{\theta}_{th}$, Note that the force provided by the sensor's parent link does not contribute to the net contact force. We adjust the threshold $\tilde{\theta}_{th}$ of these sensors to ensure that they have similar behavior to that in the real. We use a $\tilde{\theta}_{th} = 0.01N$ in simulation.

C. Benchmark Problem: In-hand Rotation

In this paper, we study the dexterity of our system by using it to solve an in-hand rotation task. In this in-hand rotation task, an object is initialized in the palm and the robot hand is then required to rotate this object around a given rotation axis.

When we are doing in-hand object rotation, the object motion is more complex than that in finger-tip rotation mentioned in section II and brings additional challenges. Specifically, the object can slide or roll in the palm during in-hand manipulation. Due to this complex motion pattern, explicit feedback from tactile or vision becomes necessary for successful manipulation. Otherwise, we are unable to infer the current state of the object and fail to push and rotate it in a secure way.

D. Discussion: What information can sensors provide?

We summarize two kinds of information our system can provide for control as follows, though its sensing is sparser than that of a real human hand.

Position information. The contact sensors can inform the policy where the object is at each time step. One example is shown in Figure 2 (left). In this example, a cuboid is placed on the palm without contacting any fingertip. At this moment, the only way to infer the position of the object is by reading the measurement of the contact sensors on the palm. This measurement can provide an estimation of the object's position (i.e., at the center), based on which the controller can decide the approximate movement of each finger, for example, driving the thumb toward the center. Without this information, the thumb can move to the right and can not come into contact with the object to initiate a rotation.

Interaction information. During in-hand object rotation, it is essential to ensure that the fingertip in charge of the
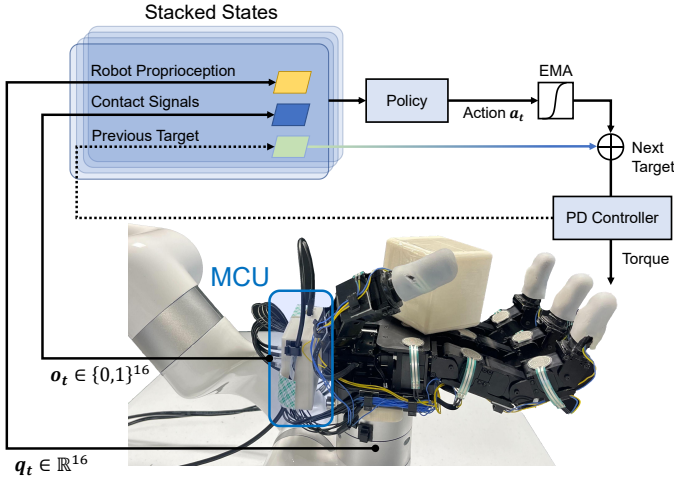
Fig. 3: Overview of the control process. The state contains tactile information, joint position, previous target, and task information like rotation axis (not shown in the figure). The policy then uses the stacked state to get the relative action, and the next target joint position is calculated. The new target is then fed to a PD controller.

rotation is indeed interacting with the object, see Figure 2 (right). Otherwise, the finger may not be able to push against the object leading to a failure, which may cause the object to move to an unstable position, and even fall out of the hand.

IV. LEARNING TOUCH DEXTERITY

A. Problem Formulation

We formulate the in-hand rotation problem as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, and $\mathcal{P}$ is the transition dynamics. $\mathcal{R}$ and $\mathcal{P}$ are unknown to the robot. The robot agent observes state s_t at each step t and take action $a_t = \pi(s_t)$ calculated by the current policy π, then it will receive a reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. The goal of the agent is to maximize the γ discounted return $\sum_{t=0}^{T} \gamma^t r_t$. The definition of these elements is as follows.

1) State: The state of the system consists of the joint position of the Allegro hand $q_t \in \mathbb{R}^{16}$, the sensor observation $o_t \in \{0,1\}^{16}$, the previous position target $\tilde{q}_t \in \mathbb{R}^{16}$, and the rotation axis $k \in \mathbb{S}^2$. Since the state at one step may not be sufficient for control, we also stack it with other 3 historical states as the input when we use an MLP as the policy network.

2) Action: At each step, the action produced by the policy network is a relative control command $a_t \in \mathbb{R}^{16}$. A PD controller then drives the hand to reach the joint position target $\tilde{q}_{t+1} = \tilde{q}_t + a_t$ at the next step. However, using this target directly may lead to non-smooth finger motion, since the actions of two consecutive steps may conflict with each other. Therefore, in practice, we use an exponential moving average as the target: $\tilde{q}_{t+1} = \tilde{q}_t + \tilde{a}_t$, where $\tilde{a}_t = \eta a_t + (1-\eta)\tilde{a}_{t-1}, t \geq 1$ and $\tilde{a}_0 = 0$. We find that $\eta = 0.8$ works well in the experiments. This PD controller operates at a control frequency of 10Hz both in the simulation and the real.
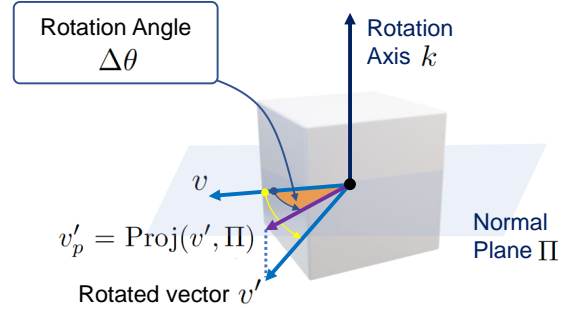

Fig. 4: Illustration of the calculation of rotation angle $\Delta\theta$: The object rotates alone Axis k and here we visualize the rotation angle $\Delta\theta$ in the Normal Plane.

3) Reward: We design a reward function that is able to make the dexterous hand rotate the object in a smooth and transferable way. The reward function used in this paper is a weighted mixture of several components:

$$r_t = w_1 r_{rot} + w_2 r_{vel} + w_3 r_{fall} + w_4 r_{work} + w_5 r_{torque} + w_6 r_{dist}. \tag{1}$$

The first term r_{rot} is the rotation reward defined as the rotated angle $\Delta\theta$ of a sampled unit vector in the normal plane Π of the rotation axis k:

$$r_{rot} = \text{clip}(\Delta\theta, -c_1, c_1). \tag{2}$$

The detailed calculation of $\Delta\theta$ is shown in Figure 4. First, we sample a unit vector v in Π randomly and we may as well imagine it is attached to the object. Then we fetch its corresponding vector v' at the next state and project it to Π: $v'_p = \text{Proj}(v', \Pi)$. $\Delta\theta \in [-\pi, \pi)$ is defined as the signed distance between v'_p and v with respect to the axis k. Note that [51] uses $\langle \omega, k \rangle$ as the rotation reward, where ω is the angular velocity returned by the simulator. Nevertheless, we find that the angular velocity provided by the simulator in our setting is very noisy since the motion of the object is very complex. As a result, using this angular velocity in the reward can usually lead to very undesirable object motion patterns, like vibrating around a specific pose. We find that using this finite difference as the reward can produce consistent rotation behavior across different runs. The second term is a penalty on the object's velocity $r_{vel} = -\|v_t\|$. This encourages the hand to rotate the object in a stable manner and increases the transferability of the trained policy. The third reward r_{fall} is a negative falling penalty when the object falls out of the palm. The fourth reward r_{work} penalize the work of controller, which is defined as $r_{work} = -\langle |\tau|, |\dot{q}_t| \rangle$. Here, τ is the outputted torque of the PD controller at step t. This penalty helps to improve the smoothness of finger motion. The fifth term $r_{torque} = -\|\tau\|$ penalizes the large torque. Finally, $r_{dist} = \text{mean}(\text{clip}(1/(\epsilon + d(x_{tip}, x_{obj})), c_2, c_3))$ is a distance reward, which encourages the fingertip to come close to the object and interact with it.

4) Reset Strategy: We design several reset strategies to reduce unnecessary exploration and speed up the learning process. First, we reset the episode when the object deviates

too much from its initial position (i.e., the center of the palm). Moreover, we reset the episode when the major axis of the object deviates too much from the rotation axis, this reduces the exploration of an undesired rotation direction.

B. Domain Randomization

We use a wide variety of domain randomization [68] to improve the Sim2real transfer.

1) Physics randomization: We randomize the object's initial position, mass, shape, and friction to ensure that the learned policy can deal with different kinds of objects.

Moreover, we randomize the gain of the PD controller to model the uncertainty of the PD controller in real. Besides, we consider randomizing each tactile sensor. For each activated contact sensor that outputs 1, with probability p we flip its output to 0. We also model the signal delay of the contact sensor by an exponential delay used in [28].

2) Non-physics randomization: We use a set of non-physics randomization to further improve the robustness of the trained policy. We inject white noises into the observation of the policy, and its outputted action to ensure that it is robust to small perturbations.

C. Training Procedure

We use the proximal policy optimization (PPO) [58] algorithm to train our control policy and multilayer perceptron (MLP) for both of the policy and value networks. We use the advantage clip threshold $\epsilon = 0.2$ and the KL threshold of 0.02. We use ELU [43] as the activation function in these networks. The policy network outputs a Gaussian distribution with a learnable state-independent standard deviation. Like [28], in order to reduce the training difficulty, we also use asymmetric observation for the policy and value network. Concretely, for the value network, we add privileged information such as the contact force over each link, the object's ground-truth pose, and physical parameters to its input. This privileged information is not accessible by the policy network. For the policy network, we only stack the current state with 3 historical states as the input.

For the IsaacGym simulation, we set $dt = 0.01667s$ with 2 simulation substeps. We use 8192 parallel environments. The action (control target) outputted by the policy network is executed by 6 steps, corresponding to a 10Hz control frequency in real.

V. EXPERIMENTS

In this part, we compare our Touch Dexterity system to several baselines in both the simulation and the real. Specifically, we are interested in the following questions:
1) *How much benefit does tactile information offer compared to the baseline in training?*
2) *Using simulation as an ideal setup, does the usage of tactile information lead to better robustness and generalization?*
3) *How well does our tactile manipulation system perform and generalize compared with other methods in the real?*
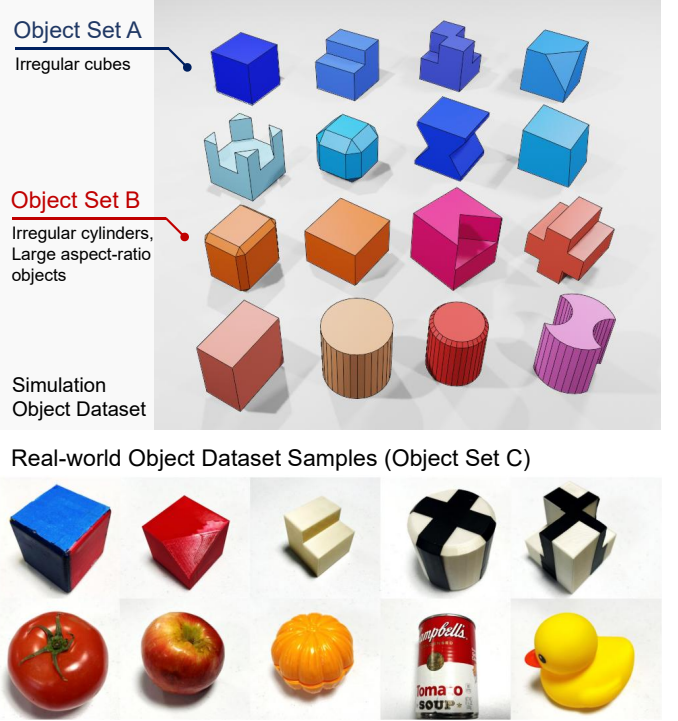

Fig. 5: The object sets used in our experiments. The full object set in the real world can be found in the supplementary material.

4) *How well does tactile perception in simulation align with that in real? How does it improve performance in real?*
We answer these questions through an extensive case study on the z-axis rotation. Then, we demonstrate that our system can also learn the rotation skill along all the other axes.

A. Experiment Setup

1) Object Dataset: For the simulation experiments, we train and evaluate our policy on a set of artificial objects of common geometries, such as cuboids, cylinders, and balls. Some examples of these objects are shown in Figure 5. Despite their simplicity, their diverse geometry can be used to approximate a large set of common daily objects. For the real experiments, we bring in some unseen real-world objects like a rubber duck, lego box for evaluation as shown in Figure 5.

2) Evaluation Metric: To evaluate the performance of a trained policy, we introduce the following metric as suggested by [51].
1) **Cumulative Rotation Reward (CRR).** We calculated the cumulative rotation reward to evaluate the rotation capability of a policy in the simulation. This metric is only used in the simulation.
2) **Cumulative Rotation Angle (CRA).** We count the cumulative rotation angle (by rounds) to evaluate the rotation capability of a policy in the real. This metric is counted by a human.
3) **Time-to-Fall (TTF/Duration).** We measure the time (by seconds) of an object staying in the palm before falling down the hand. This metric can be used both in the simulation and in real.
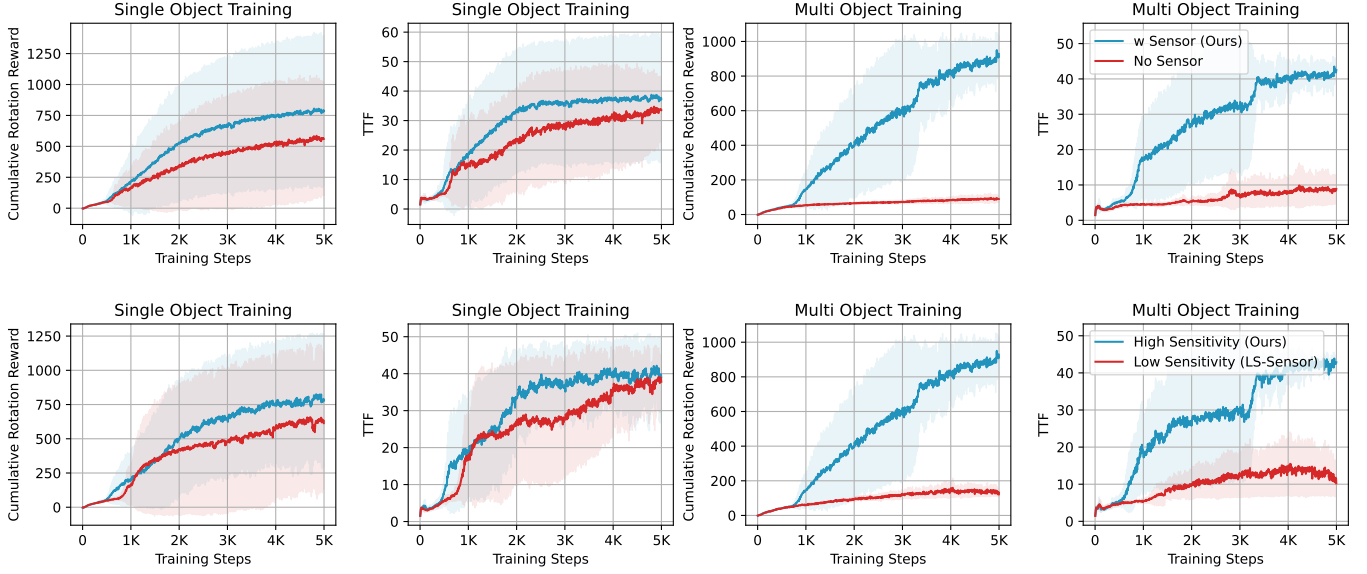
Fig. 6: Top: Policy training curve with and without sensors. Bottom: Policy training curve with sensors of different sensitivities. The results are averaged on 3 seeds. The shaded area shows the standard deviation.

B. Baselines

In the experiments, we mainly compare our methods with the following baselines.

1) **No-Sensor.** We train a PPO policy to control the hand with no tactile information available. The only way to infer the object-hand interaction information is by comparing the current joint position to the desired, target joint position. For example, when a finger (e.g. thumb) is pressing the top surface of a cuboid, we can observe a difference between these two quantities, indicating the existence of pressing behavior.

2) **LS-Sensor.** We set a higher sensor activation threshold $\theta_{th} = 0.2N$ and train another PPO policy. In other words, the sensors now have lower sensitivity to the contact, and we call this policy LS-Sensor. Under this setup, the hand is no longer able to sense some slight contacts.

3) **DS-Sensor.** This policy is used for ablation purposes. Its only difference from our policy is that it will disable all the tactile input during evaluation. This is used to test to which extent the trained tactile policy uses tactile information.

In the real-world experiments, we also introduce additional two policy baselines:

1) **Openloop Policy.** We collect several successful object rotation trajectories in the simulation and execute these trajectories on the robot. This is to study whether the considered task is complex enough.

2) **CT-Sensor.** We train a policy that uses a continuous-valued sensor input rather than the binarized version. This is to study if using continuous signals will lead to Sim2Real difficulty.

Note that there also exist some vision-based dexterous manipulation baselines like [14, 28]. However, we do not compare our method to theirs as they require a collection of a very large amount of visual simulation data, which takes significantly longer real-world time.

C. Sim: Policy learning with different sensing capabilities.

In this section, we study whether our tactile policy and the considered baseline policies are able to succeed in the training environments in the simulation. We study both the single and multi objects setup. We use the cuboid as the object in single-object training, which is common in the previous works [2, 28]. We use object set A for multi-object training. The results are shown in Figure 6. We find that in the single object setup, both the No-Sense and LS-Sensor policies have a lower rotation reward compared with our policy. Interestingly, we find that LS-Sensor can achieve a higher duration (TTF) compared with No-Sensor and can match that of our full system. This result suggests that tactile sensors of low sensitivity may still be useful to make the motion more secure. For the multi-object training, we find that our tactile policy outperforms the baseline policies by a large margin. The baseline policies fail completely in this case, while our policy is still able to succeed. This result indicates that using tactile information is essential to tame touch-only multi-object rotation. The failure of the LS-Sensor in the multi-object training case suggests that having a high-sensitivity sensor to sense the slightest contact is important.

D. Sim: Is a tactile policy robust and generalizable?

Though our tactile policy and the baseline policies can succeed in some cases during training, so far it remains unknown whether they are robust and generalizable. We consider a policy robust if it can perform well on the unseen physics parameter setup on the same set of objects. We consider a policy generalizable if it can perform well on an unseen set of objects.

TABLE I: Performance of different methods on the multi-object rotation task on the real robot. The results are averaged on 3 policies trained on 3 seeds. Each trial lasts 30 seconds. The CRA metric is measured by the number of turned rounds. The TTF metric is measured in seconds. Our proposed method can rotate both the seen and unseen objects.

Seen	Object C1		Object C2		Object C3		Object C4		Object C5	
	CRA	TTF	CRA	TTF	CRA	TTF	CRA	TTF	CRA	TTF
OL	$0.58_{\pm0.14}$	$13.30_{\pm7.77}$	$0.08_{\pm0.14}$	$4.67_{\pm8.08}$	$0.75_{\pm0.66}$	$18.67_{\pm16.29}$	$0.50_{\pm0}$	$24.00_{\pm5.29}$	$0.83_{\pm1.04}$	$13.67_{\pm15.18}$
No-Sensor	$0.25_{\pm0.25}$	$7.67_{\pm6.80}$	$0.33_{\pm0.28}$	$14.7_{\pm15.01}$	$0.08_{\pm0.144}$	$3.67_{\pm6.35}$	$0.42_{\pm0.14}$	$16.00_{\pm12.17}$	$0.25_{\pm0.25}$	$12.67_{\pm15.53}$
CT-Sensor	$2.50_{\pm3.25}$	$20.00_{\pm8.66}$	$0.75_{\pm0.66}$	$17.67_{\pm10.79}$	$2.42_{\pm2.10}$	$15.33_{\pm15.01}$	$1.92_{\pm1.46}$	$23.00_{\pm12.12}$	$1.00_{\pm0.87}$	$17.00_{\pm15.39}$
Ours	$\mathbf{4.91_{\pm0.52}}$	$\mathbf{30.00_{\pm0.00}}$	$\mathbf{2.83_{\pm1.26}}$	$\mathbf{28.67_{\pm2.31}}$	$\mathbf{2.92_{\pm1.38}}$	$\mathbf{30.00_{\pm0.00}}$	$\mathbf{4.50_{\pm1.73}}$	$\mathbf{30.00_{\pm0.00}}$	$\mathbf{2.00_{\pm0.00}}$	$\mathbf{26.67_{\pm5.77}}$

Unseen	Tomato		Apple		Orange		Soupcan		Rubber Duck	
	CRA	TTF	CRA	TTF	CRA	TTF	CRA	TTF	CRA	TTF
OL	$0.25_{\pm0.25}$	$20.00_{\pm17.32}$	$0.67_{\pm0.76}$	$20.00_{\pm17.32}$	$0.5_{\pm0.87}$	$10.00_{\pm17.32}$	$1.5_{\pm1.32}$	$20.00_{\pm17.32}$	$0.33_{\pm0.29}$	$20.00_{\pm17.32}$
No-Sensor	$0.00_{\pm0.00}$	$0.00_{\pm0.00}$	$0.33_{\pm0.58}$	$10.00_{\pm17.32}$	$0.75_{\pm1.09}$	$12.33_{\pm15.70}$	$0.08_{\pm0.14}$	$2.00_{\pm3.46}$	$0.33_{\pm0.29}$	$20.00_{\pm17.32}$
CT-Sensor	$0.33_{\pm0.29}$	$12.33_{\pm15.70}$	$0.42_{\pm0.52}$	$15.33_{\pm15.01}$	$2.08_{\pm2.10}$	$24.33_{\pm4.93}$	$2.08_{\pm2.79}$	$19.33_{\pm16.77}$	$\mathbf{1.50_{\pm0.75}}$	$\mathbf{30.00_{\pm0.00}}$
Ours	$\mathbf{1.08_{\pm0.14}}$	$\mathbf{27.33_{\pm4.62}}$	$\mathbf{2.67_{\pm1.04}}$	$\mathbf{30.00_{\pm0.00}}$	$\mathbf{3.00_{\pm1.32}}$	$\mathbf{30.00_{\pm0.00}}$	$\mathbf{4.25_{\pm1.56}}$	$\mathbf{27.33_{\pm4.62}}$	$1.42_{\pm0.38}$	$29.00_{\pm1.73}$

TABLE II: Performance of different methods on the single-object rotation task with physics distribution shift in simulation. The results are averaged on 3 seeds.

Method	Seen Physics Setup		Unseen Physics Setup	
	CRR	TTF	CRR	TTF
No-Sensor	$689.3_{\pm141.5}$	$33.3_{\pm4.7}$	$369.0_{\pm129.1}$	$23.5_{\pm6.1}$
Sensor	$\mathbf{963.8_{\pm377.8}}$	$\mathbf{42.2_{\pm4.1}}$	$919.3_{\pm338.0}$	$40.0_{\pm4.3}$
DS-Sensor	$904.2_{\pm408.6}$	$39.1_{\pm6.3}$	$615.5_{\pm293.2}$	$31.2_{\pm8.0}$
LS-Sensor	$860.0_{\pm348.7}$	$38.8_{\pm6.9}$	$796.5_{\pm366.7}$	$37.4_{\pm8.4}$

TABLE III: Performance of different methods on the multi-object rotation task in simulation. The results are averaged on 3 seeds.

Method	Seen Object Setup		Unseen Object Setup	
	CRR	TTF	CRR	TTF
Sensor	$\mathbf{976.1_{\pm86.5}}$	$\mathbf{42.1_{\pm0.6}}$	$\mathbf{594.4_{\pm63.2}}$	$\mathbf{28.2_{\pm2.7}}$
DS-Sensor	$351.5_{\pm28.0}$	$18.6_{\pm0.7}$	$186.5_{\pm16.1}$	$10.7_{\pm1.4}$

We test the robustness of the single-object setting. To do this, we sample from a smaller, unseen range of friction and mass parameters, and perform rollout. In this case, the object is more likely to slide in the hand, requiring the hand to manipulate it in a more careful manner. The results are shown in Table II. We find that there is little performance drop in our full method. However, for No-Sensor and DS-Sensor, we can observe a clear performance drop. We also find that the low-sensitivity policy LS-Sensor also performs well in the unseen physics setup. This suggests that a low-sensitivity tactile sensing ability is sufficient for the robustness of the single object rotation.

The generalization testing result is shown in Table III. We train the policies on object set A and test them on object set B. Since No-Sensor and LS-Sensor baseline does not work on the multi-object training setup, we only compare our method with DS-Sensor. We find that disabling the sensor input will lead to a significant performance drop both in the seen and unseen object setup. This result suggests that tactile information is indeed important for generalization.

E. Real: Dexterity without Vision

We have seen that our tactile policy can achieve superior performance in the ideal simulation setup. Now, we transfer the trained tactile policy to the real robot and verify if it still offers the demonstrated benefit. In real-world experiments, we train the baselines on object sets A and B and evaluate these policies on object set C. We evaluate each method using 3 different seeds for each object. The results are shown in Table I.

Our method can outperform all the baselines. It can not only perform rotation on the seen, artificial training objects but also generalize to unseen real-world objects like apples and tomatoes. The method with continuous tactile sensor signals also performs better than the other two methods without feedback. We observe that both the no-sensor and the open-loop policy can at most rotate the object for 180 degrees on the evaluated objects, after which they will get stuck or push the object off the palm, resulting in a failure. In addition, by studying the behavior of our policy and baselines, we find that our policy can adjust the finger motion immediately when objects get to positions that are easy to get stuck or fall. In contrast, the baselines without sensors do not have such kind of adaptive behavior. This result suggests that it is crucial to have a dose of tactile feedback in the considered in-hand object manipulation setup. By comparing the methods with continuous contact signals and binarized contact signals, we find that the latter has better performance. Even though the policy with continuous signals can perform well in some objects, it has poor generalizability and huge variance between different objects. This may be due to the huge gap in force measurement between simulation and the real world.
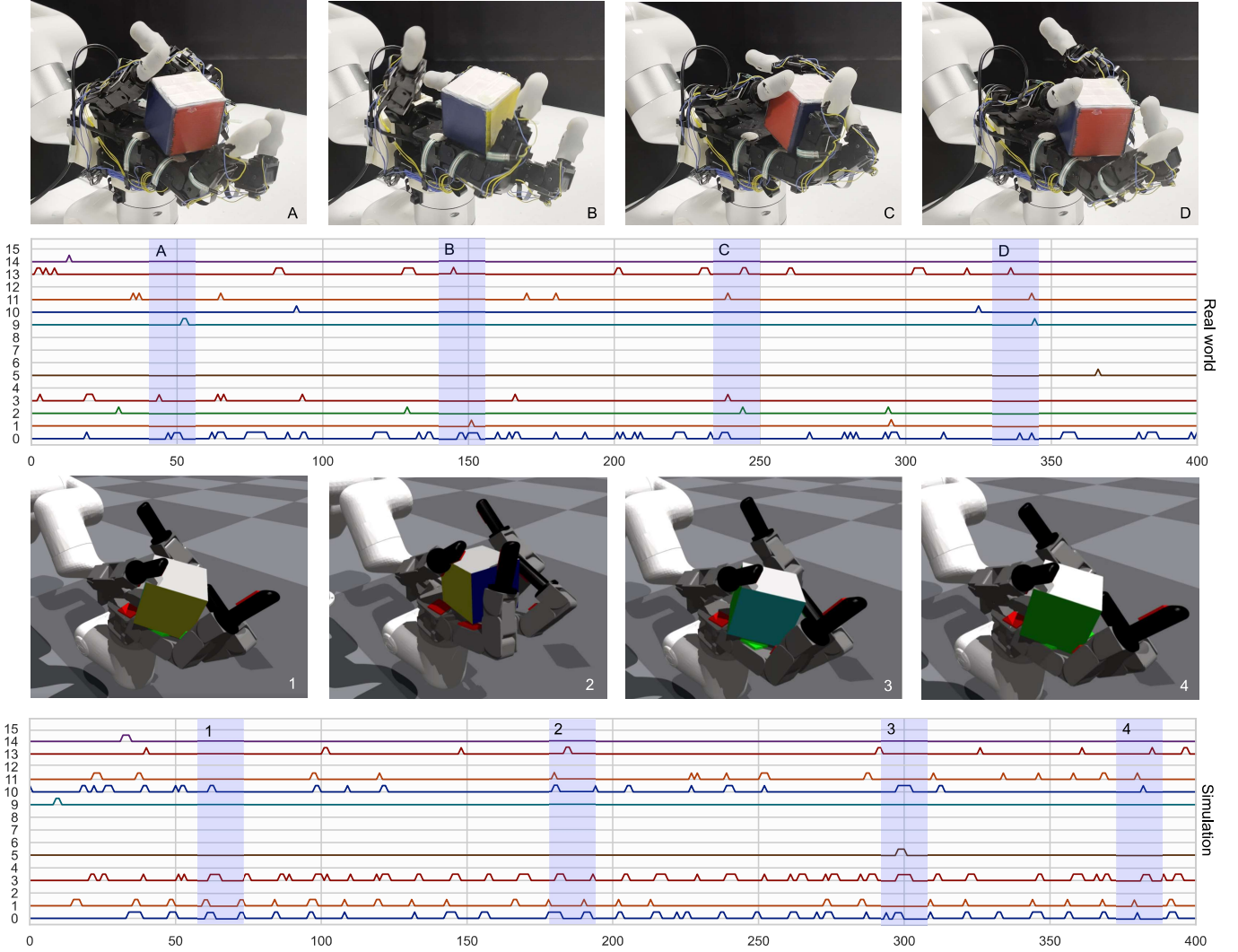
Fig. 7: Visualization of contact signals in 400 steps in the simulation and real-world experiments on a cuboid. We also show some typical frames during the rotation process. We can see that the contact signals in the simulation and the real world in general align. This accounts for successful Sim2Real transfer.

F. Qualitative Analysis: Sensor Response

To understand why Sim2Real can be successful, we conduct a case study on cuboid rotation to analyze the sensor response. We visualize two 40 seconds test trajectories recorded in the simulation and real in Figure 7 during the test. It is worthwhile mentioning that different runs in real will produce different patterns, and we put more cases in the appendix. We find that the contact signals in the simulation are slightly denser (along the temporal x-axis) and richer (along the sensor y-axis) compared with that of the real. Some sensors are also more likely to be activated (e.g., sensors 1 and 10) in the simulation, but the overall patterns of the simulation and the real are similar. This can explain why our Sim2Real transfer is successful. Moreover, when looking at local windows used by our policy (0.4s) in simulation, we observe that there are various, diverse sensor activation patterns. We hypothesize that learning from such a diverse distribution could also help the policy to transfer to the sensor observation in the real world.

G. Ablation Study I: Importance Analysis of Sensors

Then, we perform ablation studies of the system on the real robot to see which sensors are more important for a successful rotation. We divide the sensors into two groups: Fingertip and Palm. We disable these two groups of sensors and train two policies (No-Fingertip and No-Palm). Then we compare them to our full policy and DS-Sensor, see Table IV. We find that neither of the two considered policies can compare to our full policy. They achieve a similar performance as DS-Sensor, which suggests that both groups of sensors are essential for the success of in-hand object rotation.

H. Ablation Study II: A Shape Understanding Perspective

So far, we have seen that the tactile information is essential for successful object rotation. In this part, our goal is to understand its success from a shape understanding perspective. We study whether our tactile information can reveal the shape information of the object, which may be helpful for learning
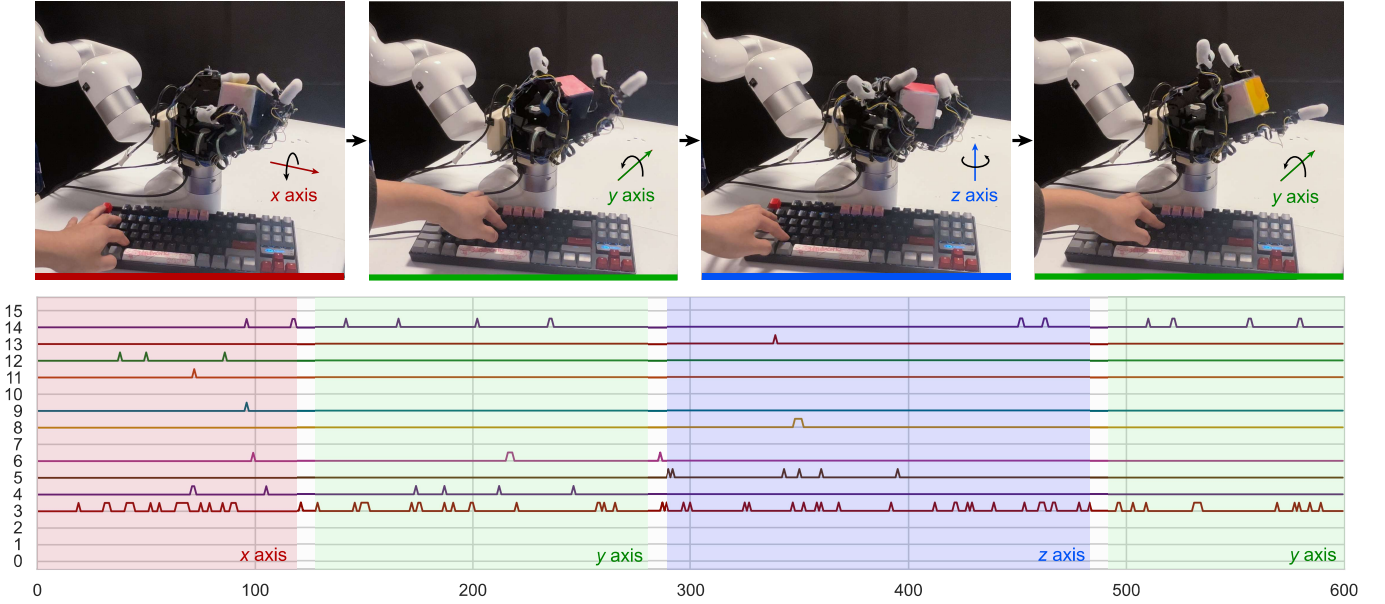
Fig. 8: With the learned rotation primitives around x, y, and z axis, we can perform human-robot shared control to reorient an object. In this example, a human operator uses a keyboard to rotate a cuboid around x, y, z, y axes consecutively. We also visualize the contact signal throughout this 600-step process (60 seconds).
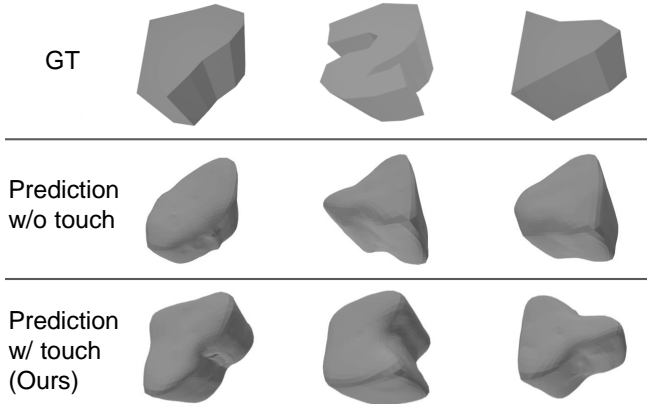

Fig. 9: Qualitative mesh reconstruction results in simulation. When the touch information does not present, we can not infer the shape of the rotated object accurately. In contrast, our method is able to reconstruct the groundtruth object by a 20-second rotation.

robust and adaptive rotation behavior across different objects. Specifically, we would like to see if it is possible to predict the shape of the object using the rollout of a rotation policy. For simplicity, we focus on the z-axis rotation of column-shaped objects. We first train a z-axis rotation policy on 125 different irregular, column-shaped objects. Then, we use this policy to collect 55000 policy rollouts of rotating these objects, and each of these rollouts lasts 200 control steps (20 seconds). Next, we split these collected rollouts into a training dataset and a test dataset. The objects in the test dataset do not present in the training dataset. We train a temporal-CNN model to

TABLE IV: Ablation analysis of the system. We train policies on different sensor setups and test their performance. The results are averaged on 3 seeds.

Method	Cuboid		Rubber Duck	
	CRR	TTF	CRR	TTF
Sensor	$\mathbf{4.91}_{\pm\mathbf{0.52}}$	$\mathbf{30.00}_{\pm\mathbf{0.00}}$	$\mathbf{1.42}_{\pm\mathbf{0.38}}$	$\mathbf{29.00}_{\pm\mathbf{1.73}}$
DS-Sensor	$0.25_{\pm0.25}$	$7.67_{\pm6.80}$	$0.33_{\pm0.29}$	$20.00_{\pm17.32}$
No-Fingertip	$0.17_{\pm0.29}$	$3.33_{\pm5.77}$	$0.42_{\pm0.14}$	$17.00_{\pm2.64}$
No-Palm	$0.42_{\pm0.38}$	$17.00_{\pm14.73}$	$0.42_{\pm0.14}$	$16.67_{\pm11.72}$

predict the shape of the object using the full rollout trajectory as input, and then we use the trained model to reconstruct the shape of objects in the test dataset. We compare our model to another ablated model, which discards all the tactile observation in the rollout during prediction (by setting them to 0). The shape reconstruction mean squared error (MSE) of our model is 0.22, while that of the ablated model is 0.45. This suggest that using tactile infomation can indeed help shape understanding. Moreover, we provide visualization of predicted object shapes are in Figure 9. With the tactile sensors, our model can reconstruct object shape much better than the ablated model. The shape understanding results suggest that the binarized tactile information is indeed important for the robot to percept the object and interact with it in a meaningful way.

I. Rotation Around Other Axes

Besides the rotation around the z axis, we also test whether our system is able to perform rotation around other axes. Here, we study the rotation around the x and y axes. To do so, we train our policy on the object set A and B as in the previous

TABLE V: Summary of rotation performance around different axes. We provide the averaged results over the object set on the x, y, and z axis rotations. The results are averaged on 3 seeds.

Rotation	Seen Obj		Unseen Obj	
	CRR	TTF	CRR	TTF
x-axis	$1.68_{\pm 0.78}$	$24.13_{\pm 6.04}$	$2.71_{\pm 1.37}$	$18.2_{\pm 9.19}$
y-axis	$1.88_{\pm 0.38}$	$22.46_{\pm 4.81}$	$1.05_{\pm 0.56}$	$23.13_{\pm 3.01}$
z-axis	$3.43_{\pm 1.22}$	$29.06_{\pm 1.45}$	$2.48_{\pm 1.27}$	$28.73_{\pm 1.34}$

experiments. The results are shown in Table V. We find that our system is still able to rotate most of the objects successfully, though it may have difficulty rotation some particular objects which results in lower CRAs. We observe that rotation around x and y axes involves many critical contacts between the object and the side of finger links. This may explain why failures can occur since the layout of our current sensor array does not support this feature. We hypothesis that a denser contact sensor array over each finger link can remedy this problem.

The rotation around x, y, and z axis provides a useful set of primitives. This enables human to use high-level commands to control the rotation behavior, as shown in Figure 8. In this example, A human operator presses the keyboard to send different rotation commands (i.e. around x, y, or z). The robot hand is then able to execute the desired rotation, reorienting the object to different poses.

VI. Conclusion

In this paper, we have presented Touch Dexterity, a new dexterous manipulation system that is able to rotate different objects through touch without vision. We showed an end-to-end reinforcement learning framework to learn dexterous manipulation skills on the proposed system. We carried out experiments both in simulation and real to demonstrate its effectiveness. Our work demonstrated that we are able to achieve touch-only dexterity as humans in real for the first time. In the future, there are many promising future directions to investigate, such as exploring the use of a more dense contact sensor array and scaling up the system to solve more diverse tasks. We hope that our work can pave the way for more intelligent robot hands.

References

[1] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1993.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research (IJRR)*, 39(1): 3–20, 2020.

[3] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

[4] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022.

[5] Yunfei Bai and C. Karen Liu. Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565, 2014. doi: 10.1109/ICRA.2014.6907059.

[6] Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. Surprisingly robust in-hand manipulation: An empirical study. In *Robotics: Science and Systems (RSS)*, 2021.

[7] Tapomayukh Bhattacharjee, Joshua Wade, and Charles C Kemp. Material recognition from heat transfer given varying initial conditions and short-duration contact. In *Robotics: Science and Systems (RSS)*, 2015.

[8] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. Reskin:versatile, replaceable, lasting tactile skins. In *Conference on Robot Learning (CoRL)*, 2021.

[9] Thomas Bi, Carmelo Sferrazza, and Raffaello D'Andrea. Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation. *IEEE Robotics and Automation Letters*, 6(3):5761–5768, 2021.

[10] A. Bicchi and R. Sorrentino. Dexterous manipulation through rolling. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 452–457 vol.1, 1995. doi: 10.1109/ROBOT.1995. 525325.

[11] Gereon Buescher, Martin Meier, Guillaume Walck, Robert Haschke, and Helge J Ritter. Augmenting curved robot surfaces with soft tactile skin. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[12] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. In *Robotics Research: The 18th International Symposium ISRR*, pages 523–539. Springer, 2020.

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

[14] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv:2211.11744*, 2022.

[15] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning (CoRL)*, pages 297–307, 2022.

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

1999.

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

[18] Alex Church, John Lloyd, Nathan F Lepora, et al. Tactile sim-to-real policy transfer via real-to-sim image translation. In *Conference on Robot Learning (CoRL)*, 2022.

[19] Siyuan Dong, Devesh K Jha, Diego Romeres, Sangwoon Kim, Daniel Nikovski, and Alberto Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[20] Neel Doshi, Orion Taylor, and Alberto Rodriguez. Manipulation of unknown objects via contact configuration regulation. In *International Conference on Robotics and Automation (ICRA)*, 2022.

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

[22] Danny Driess, Peter Englert, and Marc Toussaint. Active learning with query paths for tactile object shape exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[23] Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10598–10608, 2022.

[24] Ahsan Habib, Isura Ranatunga, Kyle Shook, and Dan O Popa. Skinsim: A simulation environment for multimodal robot skin. In *IEEE International Conference on Automation Science and Engineering (CASE)*, 2014.

[25] L. Han and J.C. Trinkle. Dextrous manipulation by rolling and finger gaiting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1998.

[26] Li Han, Yi-Sheng Guan, ZX Li, Q Shi, and Jeffrey C Trinkle. Dextrous manipulation with rolling contacts. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[27] Li Han, Yi-Sheng Guan, ZX Li, Q Shi, and Jeffrey C Trinkle. Dextrous manipulation with rolling contacts. In *International Conference on Robotics and Automation (ICRA)*, 1997.

[28] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *International Conference on Robotics and Automation (ICRA)*, 2023.

[29] Francois R Hogan, Jose Ballester, Siyuan Dong, and Alberto Rodriguez. Tactile dexterity: Manipulation primitives with tactile feedback. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[30] Roland S Johansson and Goran Westling. Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental brain research*, 56:550–564, 1984.

[31] Raj Kolamuri, Zilin Si, Yufan Zhang, Arpit Agarwal, and Wenzhen Yuan. Improving grasp stability with rotation measurement from tactile sensing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[32] Vikash Kumar, Yuval Tassa, Tom Erez, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815. IEEE, 2014.

[33] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5 (3):3838–3845, 2020.

[34] Michelle A. Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks, 2019. URL https://arxiv.org/abs/1907.13098.

[35] Qiang Li, Oliver Kroemer, Zhe Su, Filipe Fernandes Veiga, Mohsen Kaboli, and Helge Joachim Ritter. A review of tactile information: Perception and action through touch. *IEEE Transactions on Robotics*, 36(6):1619–1634, 2020.

[36] Jacky Liang, Ankur Handa, Karl Van Wyk, Viktor Makoviychuk, Oliver Kroemer, and Dieter Fox. In-hand object pose tracking via contact feedback and gpu-accelerated robotic simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6203–6209. IEEE, 2020.

[37] Xingyu Liu, Deepak Pathak, and Kris M Kitani. Herd: Continuous human-to-robot evolution for learning from human demonstration. *arXiv preprint arXiv:2212.04359*, 2022.

[38] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[39] Sami Moisio, Beatriz León, Pasi Korkealaakso, and Antonio Morales. Model of tactile sensors using soft contacts and its application in robot grasping simulation. *Robotics and Autonomous Systems*, 61(1):1–12, 2013.

[40] Artem Molchanov, Oliver Kroemer, Zhe Su, and Gaurav S Sukhatme. Contact localization on grasped objects using tactile sensing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[41] Andrew S. Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M. Dollar. Complex in-hand manipulation via compliance-enabled finger gaiting and multi-modal planning. *IEEE Robotics and Automation Letters*, 7(2):4821–4828, 2022. doi: 10.1109/LRA.2022.3145961.

[42] Adithyavairavan Murali, Yin Li, Dhiraj Gandhi, and Abhinav Gupta. Learning to grasp without seeing. In *Proceedings of the 2018 International Symposium on Experimental Robotics*, pages 375–386. Springer, 2020.

[43] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

[44] Benjamin Navarro, Prajval Kumar, Aicha Fonte, Philippe Fraisse, Gérard Poisson, and Andrea Cherubini. Active calibration of tactile sensors mounted on a robotic hand. In *Intelligent RObots and Systems Workshop on Multimodal sensor-based robot control for HRI and soft manipulation*, 2015.

[45] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, 2000.

[46] Akhil Padmanabha, Frederik Ebert, Stephen Tian, Roberto Calandra, Chelsea Finn, and Sergey Levine. Omnitact: A multi-directional high-resolution touch sensor. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 618–624. IEEE, 2020.

[47] Chaoyi Pan, Marion Lepert, Shenli Yuan, Rika Antonova, and Jeannette Bohg. Task-driven in-hand manipulation of unknown objects with tactile sensing. *arXiv preprint arXiv:2210.13403*, 2022.

[48] Austin Patel, Andrew Wang, Ilija Radosavovic, and Jitendra Malik. Learning to imitate object interactions from internet videos. *arXiv preprint arXiv:2211.13225*, 2022.

[49] Anna Petrovskaya and Oussama Khatib. Global localization of objects via touch. *IEEE Transactions on Robotics*, 27(3):569–585, 2011.

[50] Johannes Pitz, Lennart Röstel, Leon Sievers, and Berthold Bäuml. Dextrous tactile in-hand manipulation using a modular reinforcement learning architecture. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[51] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning (CoRL)*, 2022.

[52] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

[53] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7(4):10873–10881, 2022.

[54] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision (ECCV)*, 2022.

[55] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

[56] Daniela Rus. In-hand dexterous manipulation of piecewise-smooth 3-d objects. *The International Journal of Robotics Research (IJRR)*, 18(4):355–381, 1999.

[57] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[58] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[59] Jian Shi, J. Zachary Woodruff, Paul B. Umbanhowar, and Kevin M. Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017.

[60] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2361–2368, 2022.

[61] Leon Sievers, Johannes Pitz, and Berthold Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[62] Edward Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero Soriano, and Michal Drozdzal. Active 3d shape reconstruction from vision and touch. *Advances in Neural Information Processing Systems*, 34:16064–16078, 2021.

[63] Paloma Sodhi, Michael Kaess, Mustafa Mukadanr, and Stuart Anderson. Patchgraph: In-hand tactile tracking with learned surface normals. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[64] Sudharshan Suresh, Maria Bauza, Kuan-Ting Yu, Joshua G Mangelson, Alberto Rodriguez, and Michael Kaess. Tactile slam: Real-time inference of shape and pose from planar pushing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[65] Sudharshan Suresh, Zilin Si, Stuart Anderson, Michael Kaess, and Mustafa Mukadam. Midastouch: Monte-carlo inference over distributions across sliding touch. In *Conference on Robot Learning (CoRL)*, 2022.

[66] Ian H Taylor, Siyuan Dong, and Alberto Rodriguez. Gelslim 3.0: High-resolution measurement of shape, force and slip in a compact tactile-sensing finger. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[67] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey

Levine. Manipulation by feel: Touch-based control with deep predictive models. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[68] Josh 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. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2017.

[69] Pierre Tournassoud, Tomás Lozano-Pérez, and Emmanuel Mazer. Regrasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1987.

[70] Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2016.

[71] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. In *Conference on Robot Learning (CoRL)*, 2022.

[72] Jie Xu, Sangwoon Kim, Tao Chen, Alberto Rodriguez Garcia, Pulkit Agrawal, Wojciech Matusik, and Shinjiro Sueda. Efficient tactile simulation with differentiability for robotic manipulation. In *Conference on Robot Learning (CoRL)*.

[73] Jingxi Xu, Shuran Song, and Matei Ciocarlie. Tandem: Learning joint exploration and decision making with tactile sensors. *IEEE Robotics and Automation Letters*, 7 (4):10391–10398, 2022.

[74] Jianglong Ye, Jiashun Wang, Binghao Huang, Yuzhe Qin, and Xiaolong Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. *arXiv preprint arXiv:2207.05053*, 2022.

[75] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[76] Xinghao Zhu, Siddarth Jain, Masayoshi Tomizuka, and Jeroen Van Baar. Learning to synthesize volumetric meshes from vision-based tactile imprints. In *International Conference on Robotics and Automation (ICRA)*, 2022.

A. System Video Demo

We provide a video demo of our system at http://touchdexterity.github.io. The raw video demo can also be founded in the submitted files.

B. PPO Training Hyperparameters

We use the proximal policy optimization (PPO) algorithm to train our control policy. The setup of the PPO algorithm is as follows. We use an advantage clipping coefficient $\epsilon = 0.2$. We use a horizon length of 16, with $\gamma = 0.99$ and generalized advantage estimator (GAE) [57] coefficient $\tau = 0.95$. The policy network is a three-layer MLP with ELU activation. Its hidden layer is [512, 256, 256]. The policy network's learning rate is set to 1e-4, with an adaptive KL threshold of 0.02. The value network is a four-layer MLP with ELU activation. Its hidden layer is [512, 512, 256, 256]. The value network's learning rate is set to 5e-4, with an adaptive KL threshold of 0.016. We normalize the state input, value, and advantage during training. We use a gradient norm of 1.0. The minibatch size is set to 16384.

C. Improving Sim2Real Transfer

Domain Randomization We use several domain randomization techniques to improve the Sim2Real transfer. The details are shown in Table VI.

TABLE VI: Domain Randomization Setup

Object: Mass (kg)	[0.2, 0.6]
Object: Friction	[0.3, 3.0]
Object: Shape	$\times \mathcal{U}(0.95, 1.05)$
Object: Initial Position (cm)	$+\mathcal{U}(-0.015, 0.015)$
Hand: Friction	[0.3, 3.0]
PD Controller: P Gain	$\times \mathcal{U}(0.66, 1.33)$
PD Controller: D Gain	$\times \mathcal{U}(0.80, 1.20)$
Sensor: Lag Probability	0.25
Sensor: Drop Rate	0.1
Random Force: Scale	0.2
Random Force: Probability	[0.2, 0.25]
Random Force: Decay Coeff. and Interval	0.99 every 0.1s
Joint Observation Noise.	$+\mathcal{U}(-0.05, 0.05)$
Action Noise.	$+\mathcal{U}(-0.06, 0.06)$

System Identification We apply system identification to align the behavior of the PD controller in simulation to that in the real. We tune the PD coefficients to ensure that the responses of the controllers to the impulse and sinusoidal inputs are aligned. We find this step crucial for successful Sim2Real transfer.

D. Reward Design

Rotation Reward

$$r_{rot} = \text{clip}(\Delta\theta, -0.157, 0.157). \tag{3}$$

Velocity Reward

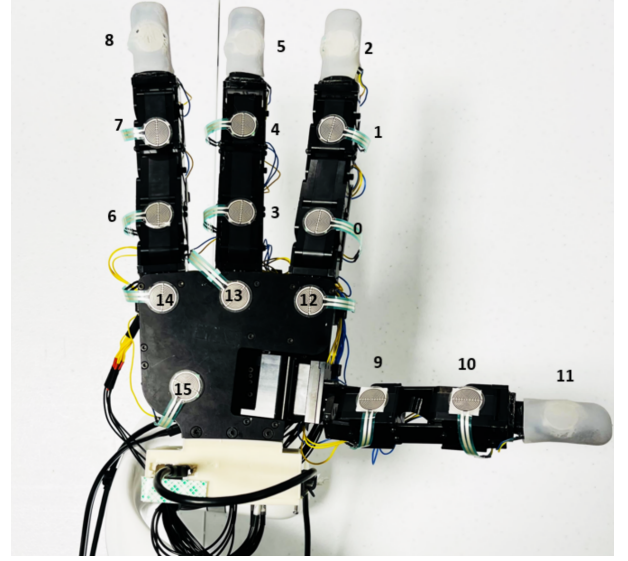$$r_{vel} = -\|v_t\|. \tag{4}$$

Fig. 10: Contact sensor map.

Falling Reward (Penalty)

$$r_{fall} = -50.0. \tag{5}$$

Work Reward (Penalty)

$$r_{work} = -\langle |\tau|, |\dot{q}_t| \rangle. \tag{6}$$

Torque Reward (Penalty)

$$r_{torque} = -\|\tau\|. \tag{7}$$

Distance Reward

$$r_{dist} = \text{mean}_{i=0,1,2,3}(\text{clip}(0.1/(0.02 + 4d(x_{tip}^i, x_{obj})), 0, 1)). \tag{8}$$

The overall reward function is

$$r_t = w_1 r_{rot} + w_2 r_{vel} + w_3 r_{fall} + w_4 r_{work} + w_5 r_{torque} + w_6 r_{dist}. \tag{9}$$

The setup of each weight: $w_1 = 20.0, w_2 = 0.1, w_3 = 1.0, w_4 = 0.0003, w_5 = 0.0003, w_6 = 0.1$.

E. More Sensor Response Examples

We show more examples of sensor activation trajectories collected in real-world experiments in Figure 11. These trajectories are collected on different objects. We can observe different activation patterns in the trajectories.
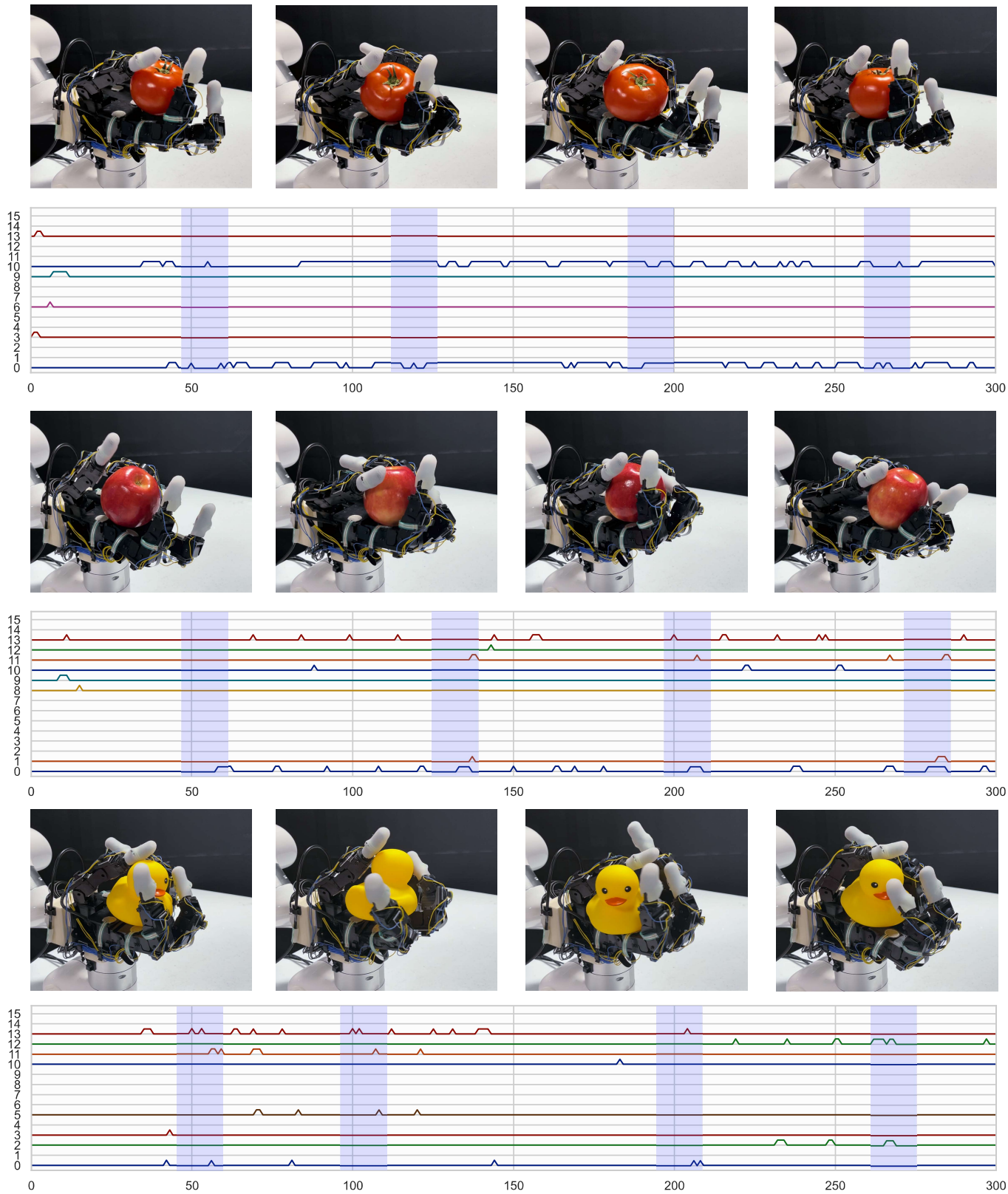
Fig. 11: More sensor activation trajectories in the real world experiments. The curves are collected on different objects and display different patterns.