

From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation

Yuzhe Qin, Hao Su[†], Xiaolong Wang[†]
UC San Diego

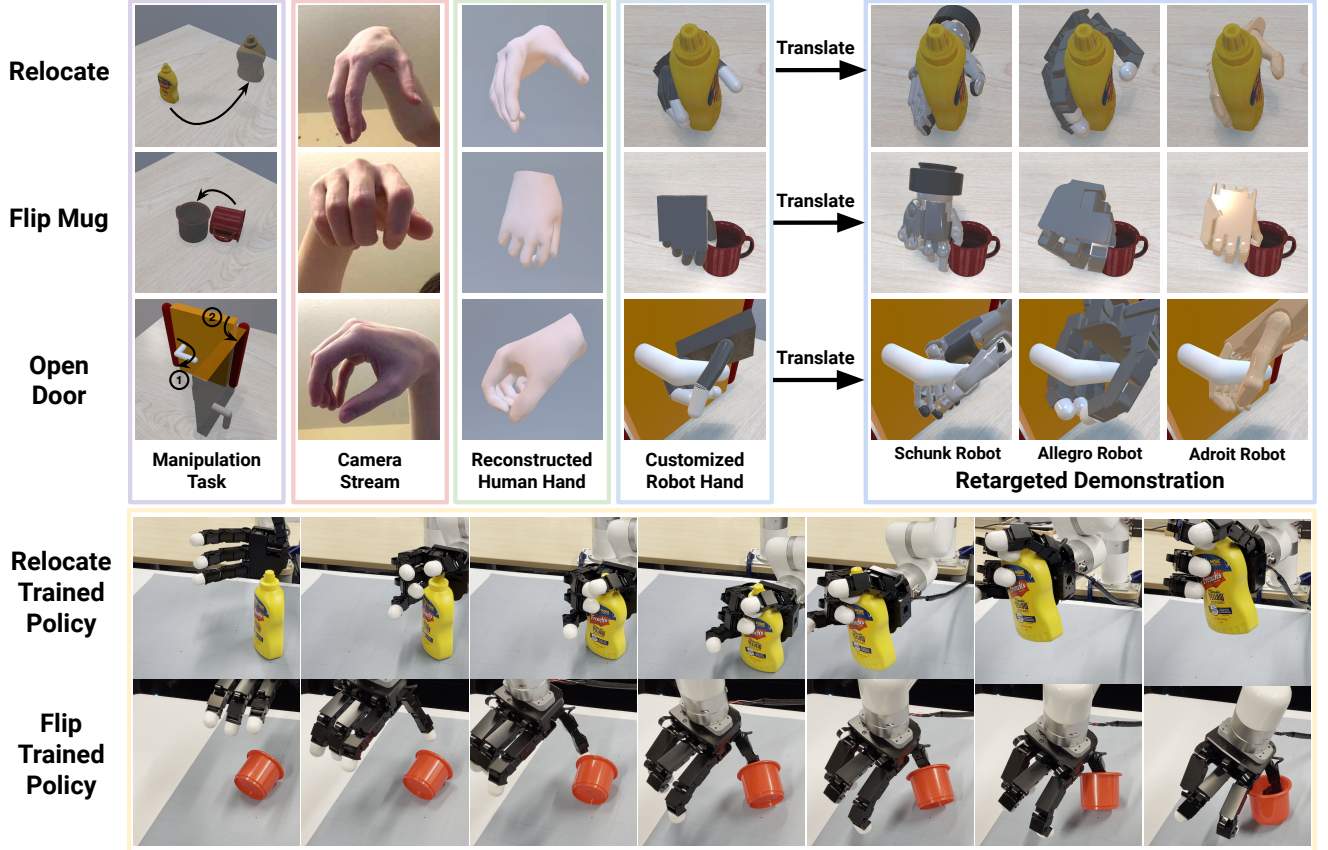


Fig. 1: **Overview:** We introduce a teleoperation system which utilizes a single camera on an iPad to stream a human hand, estimates the hand pose and shape, and converts it to a customized robot hand in a physical simulator for dexterous manipulation. Once the manipulation trajectories are collected, we translate them to different specified robot hands to generate demonstrations, and use them to perform imitation learning on the same manipulation tasks. Once the policy is trained, we deploy it to the real robot hand and show robust transfer results.

Abstract—We propose to perform imitation learning for dexterous manipulation with multi-finger robot hand from human demonstrations, and transfer the policy to the real robot hand. We introduce a novel single-camera teleoperation system to collect the 3D demonstrations efficiently with only an iPad and a computer. One key contribution of our system is that we construct a customized robot hand for each user in the simulator, which is a manipulator resembling the same structure of the operator’s hand. It provides an intuitive interface and avoid unstable human-robot hand retargeting for data collection, leading to large-scale and high quality data. Once the data is collected, the customized robot hand trajectories can be converted to different specified robot hands (models that are manufactured) to generate training demonstrations. With imitation learning using our data, we show large improvement over baselines with multiple complex manipulation tasks. Importantly, we show our learned policy is significantly more robust when transferring to the real robot. More videos

can be found in the [project page](#).

I. INTRODUCTION

Dexterous manipulation with multi-finger hand is one of the most challenging and important problems in robotics. The complex contact pattern between the dexterous hand and manipulated objects is difficult to model. It is very challenging to design a controller manually that can solve contact-rich tasks in unstructured environment. Recent research shows possibilities to learn dexterous manipulation skills with Reinforcement Learning (RL) [1]–[3]. However, the high Degree-of-Freedom (DoF) joints and discontinuous contact increase the *sample complexity* to train an RL policy.

[†]Equal advising. Correspondence to: y1qin@ucsd.edu.

Besides, black-box optimization with RL rewards can also lead to *unexpected and unsafe* behaviors.

Learning from human demonstration collected by teleoperation is a natural solution for dexterous manipulation. Most current teleoperation systems require Virtual Reality (VR) [3]–[7] devices or wired gloves to capture human hands. While providing accurate data collection, it also limits the flexibility and scalability of the process. On the other hand, vision-based teleoperation frees the human operator from wearing special devices which reduces the cost and is more scalable.

However, vision-based teleoperation introduces new challenges. It needs to convert the human hand motion into robot hand motion to command the robot, which is called motion retargeting [8]–[10]. A human operator needs to choose the next-step movement based on the imagination of the future robot hand gesture and configuration. The human operator may find it hard to control the robot if the retargeting mapping is not intuitive (e.g., controlling a robot hand with less than five fingers), and extra time will be taken to calibrate their own hands with the robot hands. Moreover, the demonstrations collected with a specific robot hand can only be used for imitation learning with the same robot.

In this paper, we introduce a single-camera teleoperation system with a scalable setup and an intuitive control interface that can collect demonstrations for multiple robot hands. Our system only requires an iPad or another mobile device as the capturing device and *DOES NOT need to perform motion retargeting online during teleoperation*. At the beginning, our system will first estimate an operator’s hand geometry (Figure 1, 3rd column in top 3 rows). The key of our system is to generate a customized robot hand on the fly in the physical simulator (Figure 1, 4th column in top 3 rows). The customized robot hand will resemble the same kinematics structure of the operator’s hand in both geometry (e.g., shape and size) and morphology. The system will generate different robot hands for different human operators, providing a more intuitive way for performing dexterous manipulation tasks efficiently.

After the demonstrations are collected with the customized robot hand, we perform motion retargeting via optimization *offline*. We convert the trajectory of a *customized robot hand* to actual *specified robot hands* (i.e., the corresponding models are manufactured and commercialized in the real world). We experiment with 3 types of robot hands including the Schunk Robot Hand [11], the Adroit Robot Hand [12], and the Allegro Robot Hand [13] (Figure 1, last 3 columns in top 3 rows). We only need to collect the trajectories once to generate imitation data for all these specified robots. Note these robot hands can even have different morphology compared to the human hand (e.g., Allegro Hand only has four fingers). We can then use the demonstrations for imitation learning on the corresponding manipulation task. We apply the imitation learning algorithm by augmenting the RL objective with the collected demonstrations in simulation [3].

We experiment with three types of challenging dexterous manipulation tasks: Relocate, Flip, and Open Door as shown

in Figure 1. By collecting data with our system using the customized robot hand, our user studies show a large advantage over previous methods on efficiency. For example, we can **efficiently collect around 60 successful demonstrations per hour** for the Relocate task, while directly operating the Allegro Hand in simulation can only collect around 10 successful demonstrations per hour. By imitation learning with the demonstrations collected by our system, we significantly improve dexterous hand manipulation on all specified robot hands over baselines in simulation.

Once the policy is learned in simulation, we can transfer it to the real robot hand. We evaluate with an Allegro Hand attached on the XArm-6 robot in the real world (Figure 1, 2 bottom rows). By incorporating human demonstrations into training, our policy learns more human-like natural behavior. Interestingly, this leads to **much more robust policy when generalizing to the real world and unseen objects**, while pure RL policy fails most of the time.

II. RELATED WORK

Dexterous Manipulation. Manipulation with dexterous robot hands has been long studied in robotics and it remains to be one of the most challenging control task [14]–[17]. Recently, we have witnessed Reinforcement Learning (RL) approaches [1], [2] delivering promising results on complex in-hand manipulation tasks. While these results are encouraging, RL suffers from poor sample efficiency in training. Under a high degree of freedom (more than 20 in most hands), the RL policy can easily explore unexpected behaviors without well-designed rewards and external constraints.

Imitation Learning from Demonstrations. Learning from human demonstrations can not only provide external constraint for the robot to explore the expected human-like behaviors but also largely reduces sample efficiency. Beyond behavior cloning [18], [19], imitation learning has been widely studied in the form of Inverse Reinforcement Learning [20]–[25] and incorporating expert demonstrations into the RL objectives [3], [26]–[29]. Our work is highly inspired by Rajeswaran *et al.* [3], where a VR setup is proposed to collect demonstrations for dexterous manipulation and an algorithm named Demo Augmented Policy Gradient (DAPG) is introduced for imitation learning. However, data collection with VR requires a lot of human effort and is not scalable. We propose to collect data via a single-camera teleoperation system, which makes the process scalable and accessible for different users. Our work is also related to imitation learning from human videos [19], [30]–[32]. However, most of these works focus on a 2-jaw parallel gripper and relatively simple tasks, where 3D information is not necessary. Our teleoperation system provides critical 3D hand-object pose information for guiding dexterous manipulation.

We emphasize our work is to train a policy that generalizes across different environment configuration, instead of training a policy to follow one expert demonstration, which is proposed in previous motion imitation literature [33]–[39].

Vision-based Teleoperation. Vision-based teleoperation frees the operator from wearing data capture devices com-

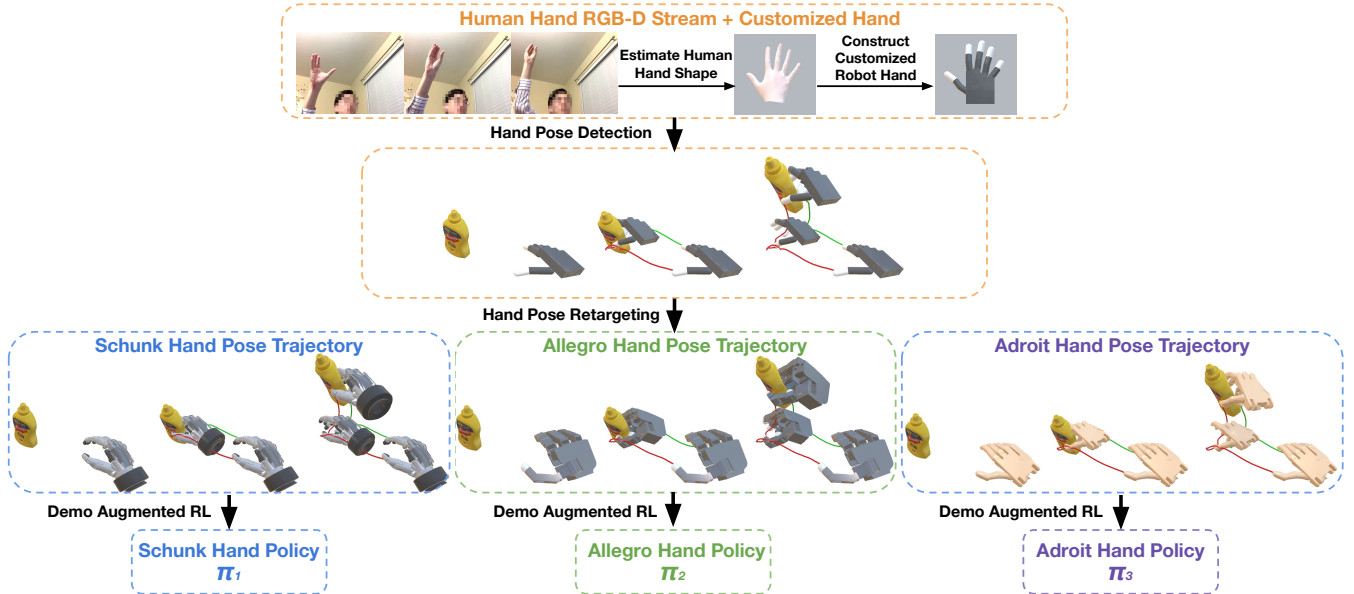


Fig. 2: **Overall Pipeline:** We stream the hand of human operator with an RGB-D camera. First we construct a customized robot hand in a physical simulator from estimated hand shape parameters result and teleoperate this robot to perform dexterous manipulation task. After teleoperation, we translate the collected trajectory on the customized hand to three different robot hands using retargeting. Finally, we train individual policy on each hand using the translated demonstrations. The red and green curve in 2nd and 3rd rows represent the finger tip trajectory of thumb and pinkie. Different box color means different hand.

monly used in game industry [40] and robot teleoperation [41], [41]–[44] on manipulation tasks, e.g. pick and place with a parallel gripper. DexPilot [8] is a pioneering work to extend the vision-based teleoperation to manipulation with an Allegro Hand [13]. To capture the hand pose, a black-clothed table with four calibrated RealSense cameras is used in their system. Our work only requires a single iPad camera for teleoperation. Our novel customized robot hand provides a more intuitive way for data collection and allows generalization for learning with multiple specified robot hands, which has not been shown before.

Concurrent Work. One concurrent work from Sivakumar *et al.* [45] performs vision-based teleoperation to control an Allegro Hand. Their work focus on mapping human hand video to robot control command, and proposed a teleoperation pipeline without using it for imitation learning. In our paper, we provides an end-to-end framework with a more intuitive teleoperation system and a paired imitation learning pipeline for policy training. Another concurrent work from Arunachalam *et al.* [46] proposes a teleoperation and visual imitating learning pipeline. They use a 2D hand pose detector during teleoperation, and propose a nearest-neighbor policy to query action from demonstration images. The pipeline proposed in our paper allows human to operate and perform more complex tasks in 3D space, and it can generalize to demonstrations for multiple robot hands.

III. OVERVIEW

We propose a novel framework in Figure 2 to learn dexterous robot hand manipulation from human teleoperation, which is composed by 3 steps as illustrated below.

(i) **Customized hand teleoperation** is proposed to collect demonstrations for dexterous manipulation tasks. It only requires video streaming from an iPad. A key innovation of

the system is constructing a customized robot hand on the fly based on the estimated shape of the operator’s own hand. The human operators can then control the customized robot hand in a physical simulation environment to perform dexterous manipulation tasks. The demonstrations can be efficiently collected with around *60 demonstrations per hour*.

(ii) **Multi-robots demonstration translation**, which can translated the original demonstration on the customized hand to any dexterous hand that is readily available in the real-world, e.g., Allegro Robot Hand. It computes the state-action trajectory, i.e. joint position and motor command, for the new specified hand that can be consumed by imitation learning algorithm. In our experiments, we try on three dexterous robot hand with different geometry, DoF, and even different number of fingers.

(iii) **Demonstration-augmented policy learning** is used to train dexterous manipulation policy on the same task as demonstrations. It augments the Reinforcement Learning objective with behavior cloning using the translated demonstration from (ii). Our framework can efficiently learn dexterous human-like skills on complex tasks which are not well solved by RL alone.

We perform Sim2Real transfer on the learned policies with a real Allegro Hand attached on the XArm-6 robot as shown in Figure 1. In our experiments, we show learning with our demonstrations can significantly increase the robustness of our policy against the Sim2Real gap.

IV. CUSTOMIZED HAND TELEOPERATION

The hardware of our teleoperation system consists of an iPad and a laptop as shown in Figure 3. We use the front camera of an iPad to stream the RGB-D video of the human operator at 25 fps. The teleoperation system consists of three components, a physical simulator, a hand detector to capture

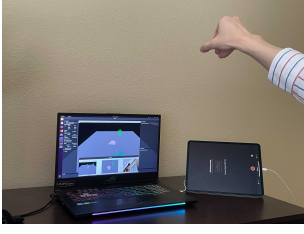


Fig. 3: Hardware setup with an iPad and a computer.

Robot	Finger DoF
Schunk	(4, 4, 3, 4, 5)
Allegro	(4, 4, 4, 4)
Adroit	(5, 4, 4, 4, 5)
Customized	(9, 9, 9, 9, 9)

TABLE I: DoF comparison for different robot models. Customized stands for the customized robot hand in [section IV](#)

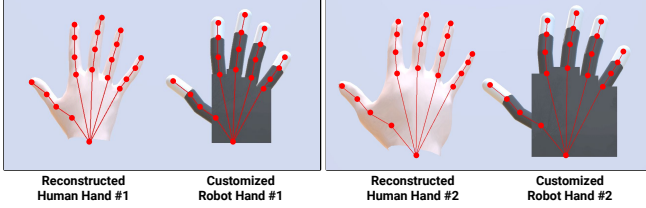


Fig. 4: Illustration of different customized robot hands generated from different human hands. The hand on left and right comes from different human. The red lines visualize the kinematics tree.

human motion, and a GUI to visualize the current simulation environment for the human operator. We use a laptop with an RTX 2070 GPU. The processing time for each RGB-D frame is less than 30ms, and the whole teleoperation system can run at 25 fps, the same as camera frequency.

A. Task Description

We construct our simulation environment on SAPIEN [47], and design multiple dexterous manipulation tasks there. The environments are used for both demonstration collection and policy learning. We develop 3 types of manipulation tasks with different objects.

Relocate. The robot picks up an object and moves it to a target position. It requires the robot to manipulate the object to match the given goal pose. The first row of [Figure 1](#) illustrates the relocate task. We experiment with three objects including *Tomato Soup Can*, *Potted Meat Can* and *Mustard Bottle* from YCB dataset [48]. It is a goal-conditioned task where we *randomize both the initial pose and the goal pose* for each trial.

Flip. As shown in the second row of [Figure 1](#), it requires the robot to flip a mug on the table. The robot needs to rotate the object 90 degrees carefully to avoid pushing the object away. This task evaluates the robot’s ability to exert force towards a certain direction. We *randomize the position and the horizontal rotation along gravity direction* of the mug for each trial.

Open Door. As shown in the third row of [Figure 1](#). The robot needs to first rotate the handle to unlock the door, and then pull the handle to open the door. The robot needs to grasp the handle with appropriate configuration so that it can achieve both the rotate and pull action. We *randomize the position* of the door for each trial.

B. Hand Detector

Our hand detector takes as input the RGB-D frames and outputs the wrist pose, hand pose parameters, and hand shape parameters. It is implemented based upon MediaPipe [49]

and FrankMocap [50]. First, we use MediaPipe hand tracker to detect the axis-aligned bounding-box and crop the image around the hand region. The cropped images are then fed into the pre-trained FrankMocap model to estimate the pose and shape parameters. Frankmocap takes the image as input and regresses the shape and pose parameters of the human hand. It can provide stable results when the hand is not in self-occlusion. We use SMPL-X [51] model to represent pose and shape parameters. It parameters the hand by shape parameters for the hand geometry and pose parameters for the deformation. Given the shape and pose parameters, we can reconstruct a hand in the canonical frame where the wrist is placed at the origin. Then we adopt the Perspective-n-Point (PnP) algorithm to match the key points in the canonical and the detected key points in the camera frame to solve the transformation of the wrist to the camera. The outputs of the hand detector to the downstream modules are wrist pose, hand pose parameters, and hand shape parameters.

C. Customized Robot Hand

Our system builds a customized robot hand based on the hand geometry of each user. Given the shape parameters from initialization, we can reconstruct a human hand at rest pose. We then build an articulated hand model in the physical simulator based on the reconstructed human hand. We extract the joint skeleton of the human hand (the red lines in [Figure 4](#)) and create a robot model with the same kinematics structure. We choose primitive shapes, e.g. box for the palm and capsules for fingers, for efficient collision detection [52] and stable simulation [53]. The customized hand has 45 (15*3) DoF, which matches the SMPL-X model. We can rotate the joints of a customized robot hand using detected pose parameters without motion retargeting. [Figure 4](#) shows different human hands and the corresponding customized hands. In this figure, the right human hand has a shorter thumb. This characteristic reflects in the customized robot hand.

We use a PD controller to control the joint angles of the customized robot hand. With each hand detection, we set the estimated pose passed by a low-pass filter as the position target. One challenge of visual teleoperation is the perception error. To tackle this issue, we utilize the hand shape estimation results as a confidence score and use it in PD control. Since the shape parameters are estimated from the best view during initialization, we can use them as the ground-truth hand shape. When the camera view is not reliable, both shape and pose estimation results will suffer from errors. Thus we can compute the error of shape parameters by comparing with the ground-truth and use it as a confidence score for pose accuracy. This problem can be formulated using normal distribution: $s_t \sim \mathcal{N}(s_0, \Sigma)$, where s_t and s_0 are the shape parameters from t frame and initialization, respectively. The covariance Σ is a diagonal matrix. We compute the normalized probability density $p(t)$ as the confidence score. The confidence-based PD position

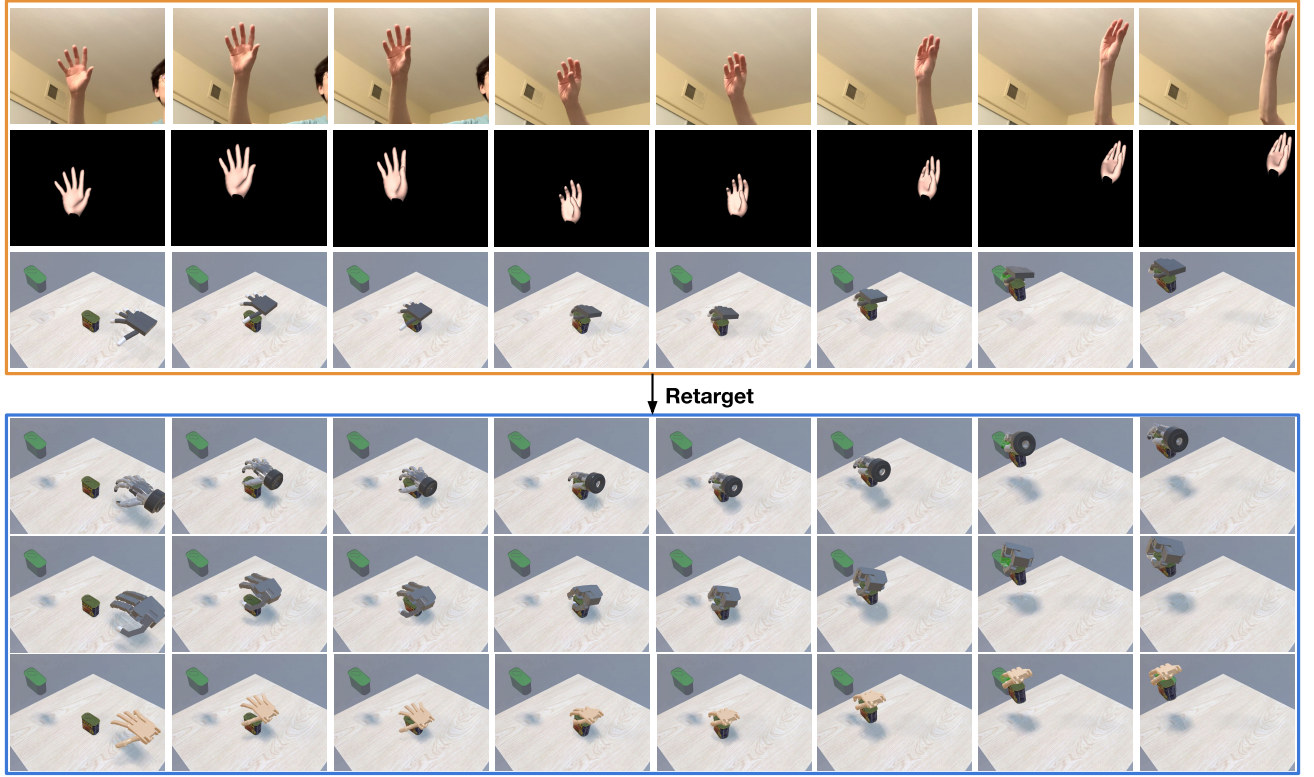


Fig. 5: **Demonstration Collection and Translation:** The top three rows show camera stream, hand pose detection results, and the teleoperated customized robot hand in simulation. The bottom three rows show the translated demonstration on three different robot hands by retargeting from the teleoperation trajectory.

control is,

$$u(t) = p(t)K_p e(t) + k_d \frac{de(t)}{dt}, \quad (1)$$

where $u(t)$ is the joint torque and k_p and k_d are PD parameters. When the perception error is large, we reduce the stiffness of controller. This design eliminates the undesired abrupt motion caused by perception error.

V. MULTI-ROBOTS DEMONSTRATION TRANSLATION

A. Hand Pose Retargeting

Table I shows the DoF of each finger for different robot models. The finger DoF is given in the order from Thumb to Pinky. We need to convert the demonstration from the customized hand to a specified robot, namely hand pose retargeting. With our customized hand, we can skip the computationally-heavy motion retargeting during teleoperation and then do it offline. We formulate the motion retargeting as an optimization problem, which is defined based on the keypoints, e.g. tip position, on the both hand as

$$\begin{aligned} \min_{q_t^R} \sum_{i=0}^N & \|f_i^C(q_t^C) - f_i^R(q_t^R)\|^2 + \alpha \|q_t^R - q_{t-1}^R\|^2 \\ \text{s.t. } & q_{lower}^R \leq q_t^R \leq q_{upper}^R, \end{aligned} \quad (2)$$

where q_t^C is joint position at step t for customized robot and q_t^R is the counterpart for the specified robot, e.g. Schunk Robot Hand. We use f_i^C and f_i^R to represent the forward kinematics function of i -th keypoints on the two robots. To improve the temporal consistency, we add a normalization

term to penalize the joint position change and initialize q_t^C using the value from q_{t-1}^C . After solving the objective above, we can compute the joint position trajectory q_t^R for any specified robot. We apply the hand pose retargeting individually for each specified robot in Figure 2.

B. Action Computation

Hand motion retargeting convert the joint pose trajectory from customized hand to another dexterous hand. To support demo-augmented policy learning, we also need the action for each finger joint. We follow the action estimation procedure in DexMV [54] to compute the action, i.e. joint torque or motor control command, from joint pose trajectory of the specified robot. We first pass the joint pose trajectory into a first-order low-pass filter. Then we compute the joint torque via manipulator equation [55] of robot inverse dynamics $\tau = f_{inv}(q, \dot{q}, \ddot{q})$. For more details, please refer to [54].

Visualization on Teleoperation and Translation. We use Figure 5 to summarize the previous two sections. The first three rows show the human hand controlling a customized robot hand in simulator to relocate a can object to a target position (a transparent green shape). Once the demonstration is collected, we can convert it to demonstrations for different robot hands (Schunk, Allegro, Adroit hands) executing the same task in the bottom three rows.

VI. DEMONSTRATION-AUGMENTED POLICY LEARNING

Given the retargeted demonstration, we perform imitation learning to solve the dexterous tasks defined in subsection IV-

Robot	S.1 success	S.1 Time	S.2 success	S.2 Time
Schunk	61.2%	14.2s	30.6%	30.3s
Adroit	58.8%	11.5s	28.2%	37.6s
Allegro	44.7%	18.7s	16.9%	42.5s
Customized	78.9%	9.1s	55.3%	23.0s

TABLE II: Success rate and completion time for *Relocate* task. S.1 and S.2 denotes stage 1 and stage 2.

Robot	S.1 success	S.1 Time	S.2 success	S.2 Time
Schunk	83.5%	9.2s	60.0%	20.4s
Adroit	81.2%	8.5s	61.2%	18.9s
Allegro	71.8%	12.7s	41.1%	23.6s
Customized	95.3%	6.2s	82.4%	15.3s

TABLE III: Success rate and completion time for *Open Door* task. S.1 and S.2 denotes stage 1 and stage 2.

A. Naive behavior cloning may be hard to work with randomized initial and target pose. Instead, we adopt imitation learning algorithms that incorporate the demonstration into RL. Specifically, we use Demo Augmented Policy Gradient (DAPG) [3] formulated below as our imitation algorithm.

$$g_{aug} = \sum_{(s,a) \in \rho_{\pi_{\theta}}} \nabla \ln \pi(a|s) A^{\pi}(s,a) + \sum_{(s,a) \in \rho_{\pi_{demo}}} \nabla \ln \pi_{\theta}(a|s) \lambda_0 \lambda_1^k \max_{(s',a') \in \rho_{\pi}} A^{\pi}(s',a'),$$

where the first line is the vanilla policy gradient objective in RL and the second line is imitation objective using demonstration. It can be regarded as a combination of behavior cloning and online RL. ρ_{π} is the occupancy measure under policy π , λ_0 and λ_1 are hyper-parameters, and k is the training iterations. $A^{\pi}(s',a')$ is the advantage function.

VII. EXPERIMENT

We first demonstrate the benefits of using the proposed customized robot hand in teleoperation for data collection by a user study with 17 different human operators. Then we show that by leveraging the demonstrations collected by our system, we can improve the policy learning performance by a large margin on various tasks in simulated environment. Finally, we perform real-world experiments on the Allegro hand attached on the X-Arm 6, which shows that the demonstration can improve the policy robustness when transferring to the real-world with higher success rate.

A. Teleoperation User Study

We compare the proposed customized robot hand with the standard robot hand during teleoperation. We ask 17 different human operators to perform *Relocate* and *open door* tasks using 4 different robot hand models: (1) Customized robot hand; (2) Schunk SVH hand; (3) Adroit hand; (4) Allegro hand. For the last three robots, online motion retargeting is required to convert human hand motion onto robot motion while for the customized robot hand we directly use the human pose parameters as the PD target for each joint.

Task Setup. Each human operator is asked to perform *Relocate* and *open door* with all four robot hands. Each task-robot pair is tested five consecutive times. For *Relocate* task, the randomly-sampled target position is visualized by

a transparent-green shape, as shown on the top-right of Figure 2. For each task, the operator will have three-minute trials to get familiar with the task. A common issue is that operators will become more proficient during the testing. They tend to get better results for the task-robot pairs tested later than the former one. For fairness, the order of robot hands to be tested is randomized for each operator.

Evaluation Protocols. We divide both *Relocate* and *open door* tasks into two stages. For *Relocate*, the first stage is succeeded when the object is lifted up while the second stage is succeeded when the distance between object and target is smaller than a given threshold. For *open door*, the first stage is successful when the door is unlocked by rotating the handle while the second stage is succeed when the door is opened. We will report the average success rate and completion time for each stage of each task. Note that the completion time does not include the time for initialization, which is required for all these four robots to construct the frame alignment between simulated robot hand and real human hand.

Results. The average success rate and task completion time over all operators are shown in Table II for *Relocate* and Table III for *Open Door*. The customized robot hand achieves the highest success rate on all tasks with a large margin compared with the online retargeting method on the other three hands. Considering the initialization process and other overhead, operator can collect around 60 demos per hour for *Relocate* while using allegro hand can only get 10 demos with success. Human operators report that the customized hand is more controllable than other robot hands. One cause is the uncontrollable time consumption required by online motion retargeting. On the laptop with specified in section IV, the motion retargeting steps will takes 76 ± 65 milliseconds. The large variance is caused by iterative optimization in online retargeting. It increases the difficulty of predicting next-step hand motion. By removing the online retargeting using our customized hand, the teleoperation system can provide smoother and more immediate feedback to human operators. We also find the allegro hands perform the worst in most metrics. One possible cause is that allegro hand only has 4 fingers, and it is much larger than other robot hands with 253 mm length, while the average length is 193mm for adult males and is 172mm for adult females.

B. Task Learning Comparison

We evaluate on the tasks of *Relocate* three different objects, *Flip* a mug, and *Open Door*. We use the processed demonstration to train policy to perform these tasks and compare them with the RL baseline. We ablate how friction, PD Controller Parameters, Object Density, and the number of demonstrations can affect the learning process. For the RL baseline, we use Trust Region Policy Optimization (TRPO) [56] as the on-policy algorithm. Both policy and value function are 32×32 2-layer Multi-Layer Perceptrons (MLPs). The TRPO will use 200 trajectories for each step. The imitation learning algorithm is DAPG described in section V with the same hyper-parameters as TRPO. We

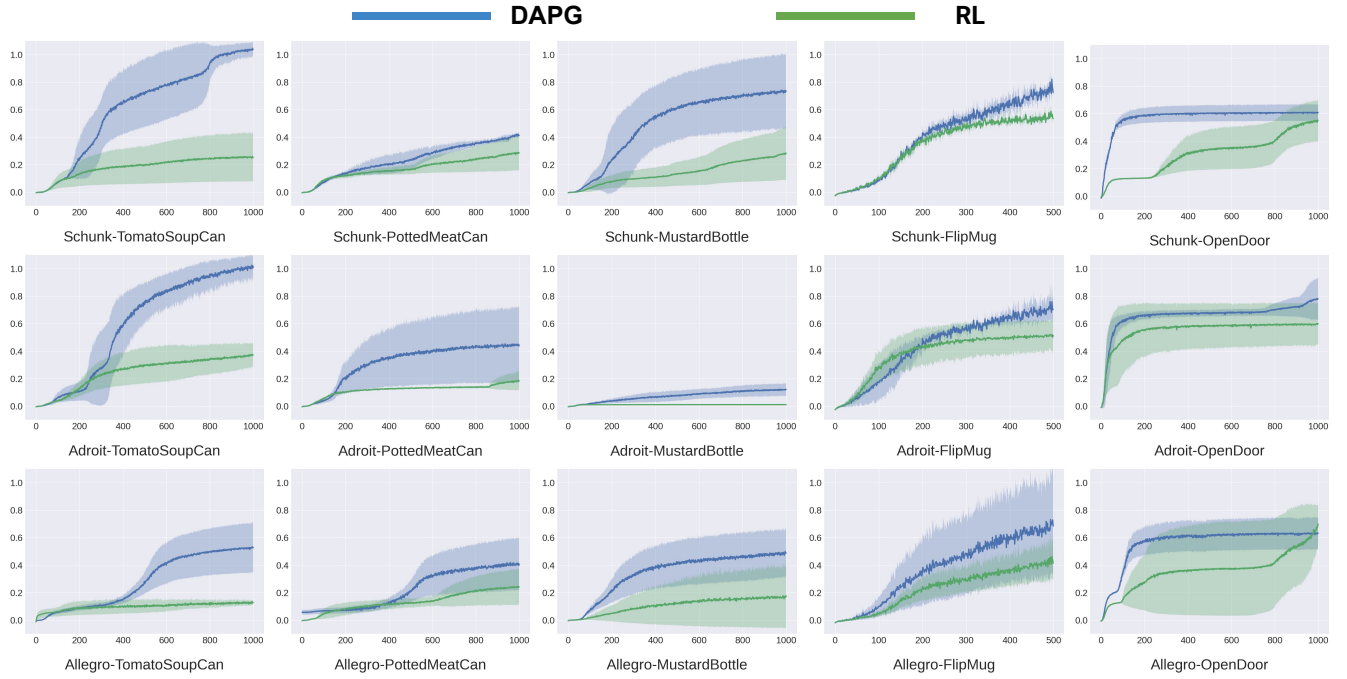


Fig. 6: Learning curves of RL and DAPG on all tasks with four different robot hands in four rows. The first three columns are Relocate task with three objects. Following two columns are Flip Mug and Open Door tasks. The x-axis is training iterations and y-axis is the normalized return. The shaded area indicates the standard deviation for three random seeds.

Task	RL	DAPG
Relocate-Toma.	45.3 ± 4.0	85.0 ± 12.3
Relocate-Pott.	6.7 ± 6.3	41.0 ± 20.3
Relocate-Must.	44.0 ± 20.0	75.3 ± 24.7
Flip-Mug	48.0 ± 4.3	77.3 ± 5.0
Open-Door	55.0 ± 14.3	69.0 ± 7.7

Schunk Robot

Task	RL	DAPG
Relocate-Toma.	41.7 ± 30.3	95.0 ± 3.0
Relocate-Pott.	0 ± 0	53.3 ± 37.7
Relocate-Must.	0 ± 0	0 ± 0
Flip-Mug	28.7 ± 17.7	54.7 ± 15.3
Open-Door	58.3 ± 21.7	78.0 ± 19.7

Adroit Robot

Task	RL	DAPG
Relocate-Toma.	0 ± 0	59.7 ± 21.3
Relocate-Pott.	4.3 ± 3.7	38.3 ± 21.7
Relocate-Must.	36.3 ± 15.3	49.7 ± 18.3
Flip-Mug	33.7 ± 15.0	51.3 ± 34.7
Open-Door	69.3 ± 38.0	64.7 ± 14.7

Allegro Robot

TABLE IV: Success rate of the evaluated methods. We use \pm to represent mean and standard deviation over three random seeds. Relocate task has three different objects: tomato soup can, potted meat can, and mustard bottle. The success of *Relocate* is defined based on the distance between object and target. The success of *Flip* is defined based on the orientation of the object. The success rate of *Open Door* is defined based on the joint angle of door hinge.

collect 50 trajectories of demonstration for each task and retarget the motion from customized hand to the specified robot. We train policies with three different random seeds.

The robot state space contains robot joint angles, velocity of hand palm, object position, and orientation at each time step. We include target position for *Relocate* and joint angle of door for *Open Door*. The action space is composed of two parts: hand palm and finger joints. The motion of the palm is controlled by 6 velocity controllers (3 for translation, 3 for rotation). And the finger joints are actuated by PD position controllers.

We evaluate both RL and DAPG on *Relocate*, *Flip*, and *Open Door* tasks. The training curves are shown in Figure 6. The success rate of three specified robot hand is summarized in Table IV. For *Relocate*, the task is considered successful when the object position is within 0.1 unit length to the target at the end of the episode. For *Flip*, the robot will get success when the orientation of mug is flipped back, where the angle between the negative z-axis and the direction of gravity is less than 5 degree. For *Open Door*, the task is successful when the joint angle of door hinge is larger than 60 degrees.

As shown in Figure 6 and Table IV, imitation learning method outperforms the RL baseline for most tasks and robots. For most robots on all tasks, DAPG can outperform pure RL, which shows the demonstration generated by motion retargeting can improve policy training. The only exception is *Open Door* with allegro hand. We visualize the policy trained by DAPG and RL in Figure 8: DAPG tries to open the door by grasping the handle in a natural behavior while RL policy press on the handle with a large force and open the door purely by friction. These results highlight the value of demonstration to regulate the behavior of policy to be expected (human-like) and safe. We find that for both RL and DAPG, *Relocate* with a mustard bottle using Adroit and Allegro robot is very challenging. The reason is that the thumb and other fingers can not form a tight shape closure. While for the Schunk robot, the freedom of the thumb is large enough to grasp the object. On the other hand, Adroit achieves the best on *Relocate* with a tomato soup can. This indicates the existence of robot-specific skills. Different robot hands are designed to fit objects with different geometry and a single robot hand can hardly do best for all tasks.

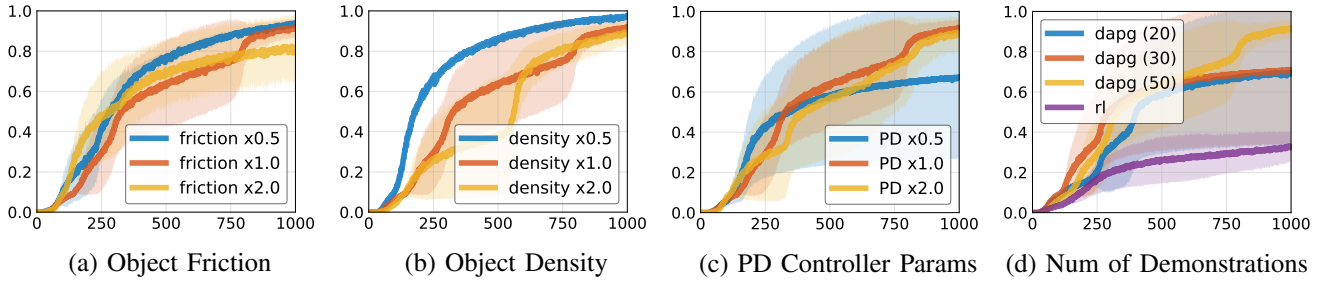


Fig. 7: **Ablation Study:** Learning curves of DAPG on the *Relocate* task with tomato soup can using Schunk Robot Hand. We ablate: (a) friction parameter of the relocated object; (b) density of object; (c) PD controller parameters: stiffness and damping; (d) number of demonstrations used to train DAPG. The demonstrations are kept the same for all conditions.

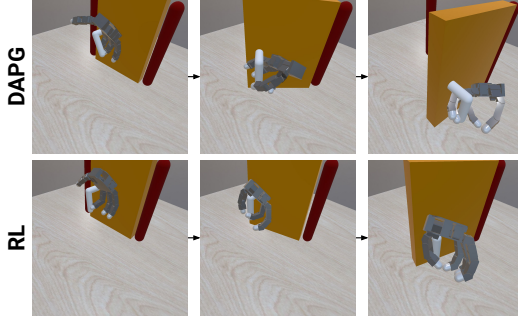


Fig. 8: Comparison of the naturalness on *Open Door* using Allegro Robot Hand. **Top Row:** policy learned by DAPG with demonstrations; **Bottom Row:** policy learned by RL without demonstrations.

C. Ablation Study

To investigate the influence of different dynamics conditions and the number of demonstrations, we ablate the object friction, controller parameters, object density, and the number of demonstrations. We choose the *Relocate* task with tomato soup can using Schunk robot for ablation study. Figure 7 (a) shows that the learning curve is robust to friction change. To hold the object firmly, a two-finger parallel-jaw gripper usually needs to form an antipodal grasp [57], which is sensitive to friction change. Different from parallel-gripper, the dexterous hand can form force closure with multiple contact points, thus can withstand smaller friction. Similar results can also be found in Figure 7 (b). Figure 7 (c) illustrates the influence of controller parameters. With larger stiffness, the robot can move the object to the target sooner and get a larger reward while controllers with smaller PD can still solve the task. Figure 7 (d) shows more demonstrations can achieve better performance. We also observe that when using only 20 or 30 demos, the variance is larger.

D. Real-World Robot Experiments

In the real-world robot experiments, we attached an Allegro hand onto a XArm-6 robot arm [58] instead of using a flying hand. The experiment setup is shown in Figure 9. We evaluate on the *Relocate* and *Flip* tasks. In simulation, we also build the same XArm6+Allegro model as real-world robot. Similarly as the experiments for flying hand in subsection VII-B, we use the motion retargeting method in subsection V-A to generate demonstrations and train RL and imitation learning policies in simulation. To facilitate

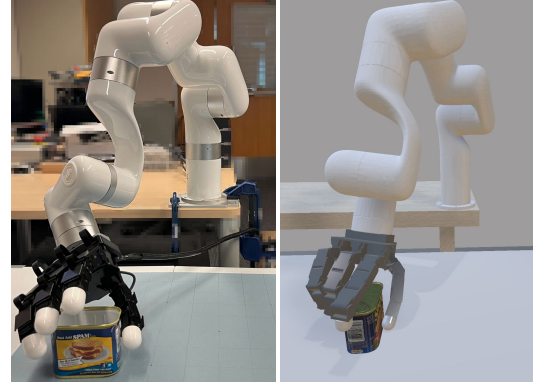


Fig. 9: **Left side:** real robot setup. The cyan poster on the table is a reference coordinate to determine whether the object is moved to the target position. **Right side:** simulated robot arm setup.

the sim2real transfer, we apply additive Gaussian noise onto the object pose to the observation and randomize the dynamics parameters, such as friction, density, and controller PD parameters during policy training.

Task Setup. The observation space are composed of robot proprioceptive state, object pose. The target object position is additionally included for *Relocate*. The object pose in the observation is fixed during the episode and only the initial pose is given, which is estimated by Iterative Closet Point (ICP) algorithm using the point cloud captured by a RealSense D435 camera.

For *Relocate*, we randomize the initial and target object position for each evaluation trail. The initial object position is randomized within a 20cm square and the target object position is randomized within a 40cm square with fixed height. The task is success if the robot can lift the object up to the target position. To determine the final object position, we use the reference coordinate on the table as shown in Figure 9 to record the xy position of object. If the difference of xy position between object and target is within 5cm, the trail is considered as a success. In the experiments, we divide the objects into two groups: known object and novel objects. As visualized in Figure 10, the known object group is composed of three objects that the policies are trained on while the novel object group is composed of five objects that are not seen during training. We evaluate the policy separately on both groups. The task execution sequence is visualized on the top two rows of Figure 11.



Fig. 10: **Relocate Task**: Visualization of known objects and novel objects used in our experiments. The first row shows the grasping process and the second row show the object. We test on three known objects and five novel objects that not presented during training.

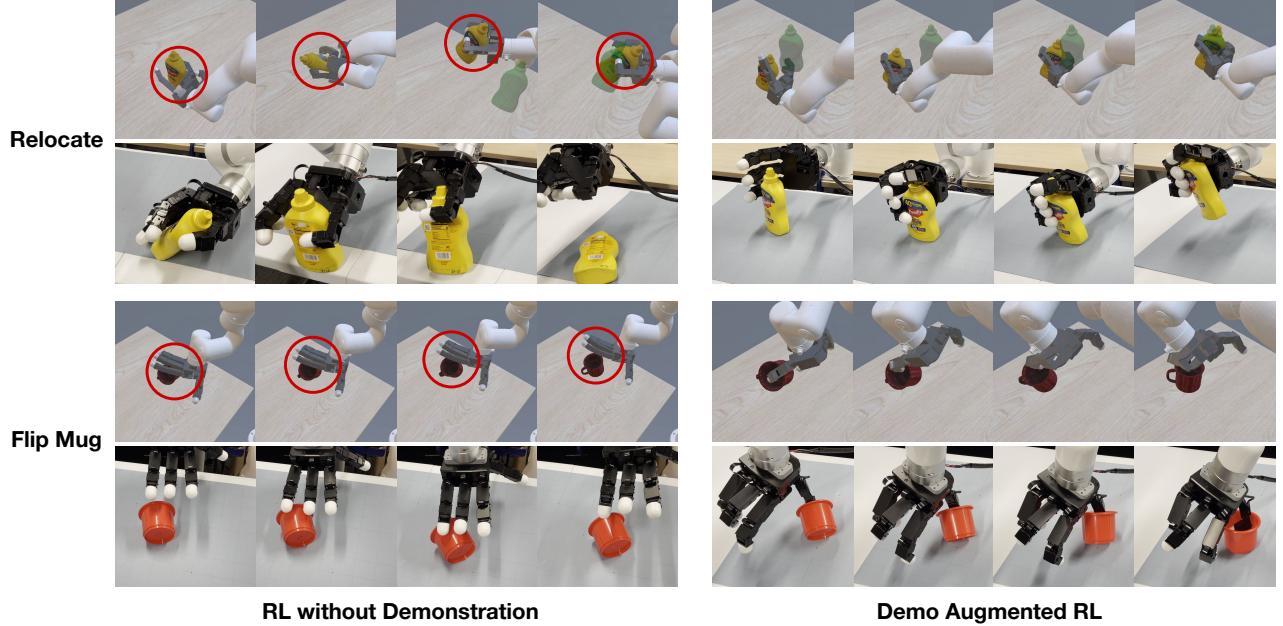


Fig. 11: Policy comparison between pure RL and the demo augmented RL trained with our demonstrations. The RL policy can get success in simulation, but due to its unstable behavior, it can hardly transfer to the real world.

For *Flip*, we randomize the initial object position within a 20cm square. The task is success if the distance between the table top and the highest point on the bottom of mug is smaller than 1cm, which means that the orientation of mug is nearly vertical. The task execution sequence is visualized on the bottom two rows of Figure 11.

Quantitative Results. During evaluation, we randomly sample 9 object initial and target position pairs and use the same pairs for each policy. For both known object and novel object settings in the Relocate task, we also sampled the object randomly and use the same set of sampled objects for each policy. We evaluate both RL and DAPG policies trained with three different random seeds. The success rate for both tasks is shown in Table V. We find when transferred to the real robot, the gap between imitation learning and pure RL is much larger than it is in simulation. We conjecture that a more human-like manipulation policy with imitation learning is more robust to the Sim2Real gap. More interestingly, for the Relocate task, the learned policies can even generalize to novel objects that are not seen in training. Note in our experiments, the geometric shape is not captured by the

Task	RL	DAPG
Relocate-Known.	22.2 ± 22.2	77.7 ± 11.1
Relocate-Novel.	18.5 ± 23.1	66.6 ± 11.1
Flip Mug	3.6 ± 6.4	44.4 ± 19.2

TABLE V: Success rate of the evaluated methods on Relocate and Flip tasks. We use \pm to represent mean and standard deviation over three random seeds.

policy, but only the 6D object pose is. This shows the advantage of multi-finger hand: When operating like human, it offers a certain robustness against the change of shape.

Policy Visualization in Simulation and Real. To illustrate why the imitation learning policy is more robust to the physics gap in Sim2Real, we provide visualizations for both tasks in Figure 11. On the top two rows for the Relocate task, we observe that both RL and imitation learning can successfully achieve the task in simulation. However, the RL policy tends to grasp the object using only two fingers with unstable contact (highlighted by red circle). In contrast, policy trained with demonstrations use all four fingers. This leads to a big difference for the real robot: The object slip down from the hand for RL policy while imitation learning

policy can stably grasp the object.

The bottom two rows of Figure 11 are the Flip task. Pure RL policy solve the task by pushing on the mug swiftly in simulator while imitation learning policy first place one finger inside the mug and then rotate the wrist. The pushing action from RL policy can hardly success with the real robot hand. While the behavior close to human demonstrations achieve much better results in the real world.

From both examples, we observe that learning with pure RL for multi-finger hands will try to hack the physics in the simulator and achieve unnatural behaviors, which is hard to transfer to the real world. One the other hand, learning human-like behavior using imitation learning with our demonstrations allows much more robust and stable policy for real world application.

VIII. CONCLUSION

We propose a novel single-camera teleoperation system to collect human hand manipulation data for imitation learning. We introduce a novel customized robot hand, providing a more intuitive way for different human operators to collect data. We show the collected demonstrations can improve the learning of dexterous manipulation on multiple robots and robustness when deployed in real world, when the data collection only needs to be conducted once.

REFERENCES

- [1] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *arXiv*, 2018.
- [2] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv*, 2019.
- [3] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *RSS*, 2018.
- [4] V. Kumar and E. Todorov, "Mujoco haptix: A virtual reality system for hand manipulation," in *International Conference on Humanoid Robots (Humanoids)*, 2015.
- [5] H. Hedayati, M. Walker, and D. Szafrin, "Improving collocated robot teleoperation with augmented reality," in *International Conference on Human-Robot Interaction*, 2018.
- [6] Y. Pan, C. Chen, D. Li, Z. Zhao, and J. Hong, "Augmented reality-based robot teleoperation system using rgb-d imaging and attitude teaching device," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102167, 2021.
- [7] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.
- [8] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *ICRA*, 2020.
- [9] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang, "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *ICRA*, 2019.
- [10] D. Antotsiou, G. Garcia-Hernando, and T.-K. Kim, "Task-oriented hand motion retargeting for dexterous manipulation imitation," in *ECCV Workshops*, 2018.
- [11] Schunk, "Schunk svh robot hand," <https://schunk.com/us/en/gripping-systems/highlights/svh>.
- [12] V. Kumar, Z. Xu, and E. Todorov, "Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands," in *ICRA*, 2013.
- [13] W. Robotics, "Allegro robot hand," <https://www.wonikrobotics.com/research-robot-hand>.
- [14] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-d objects," *The International Journal of Robotics Research*, 1999.
- [15] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *ICRA*, 2000.
- [16] S. Andrews and P. G. Kry, "Goal directed multi-finger manipulation: Control policies and analysis," *Computers & Graphics*, 2013.
- [17] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," 2014.
- [18] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NeurIPS*, 1989.
- [19] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," *arXiv*, 2020.
- [20] A. Y. Ng, S. J. Russell, et al., "Algorithms for inverse reinforcement learning," 2000.
- [21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," 2004.
- [22] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, 2016.
- [23] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv*, 2017.
- [24] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv*, 2018.
- [25] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. de Freitas, "Playing hard exploration games by watching youtube," in *NeurIPS*, 2018.
- [26] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, 2008.
- [27] Y. Duan, X. Chen, R. Houthoofd, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," 2016.
- [28] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv*, 2017.
- [29] I. Radosavovic, X. Wang, L. Pinto, and J. Malik, "State-only imitation learning for dexterous manipulation," *IROS*, 2021.
- [30] K. Schmeckpeper, O. Rybkin, K. Daniilidis, S. Levine, and C. Finn, "Reinforcement learning with videos: Combining offline observations with interaction," *arXiv*, 2020.
- [31] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2robot: Learning manipulation concepts from instructions and human demonstrations," in *RSS*, 2020.
- [32] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *Robotics and Automation Letters*, 2020.
- [33] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfvr: Reinforcement learning of physical skills from videos," *TOG*, 2018.
- [34] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *ICRA*, 2018.
- [35] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *ICLR*, 2018.
- [36] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, "Multiple interactions made easy (mime): Large scale demonstrations data for imitation," *arXiv*, 2018.
- [37] G. Garcia-Hernando, E. Johns, and T.-K. Kim, "Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning," *arXiv*, 2020.
- [38] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-structured visual imitation," in *CoRL*, 2020.
- [39] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg, "Learning by watching: Physical imitation of manipulation skills from human videos," *arXiv*, 2021.
- [40] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.
- [41] J. Kofman, S. Verma, and X. Wu, "Robot-manipulator teleoperation by markerless vision-based hand-arm tracking," *International Journal of Optomechatronics*, 2007.
- [42] G. Du, P. Zhang, J. Mai, and Z. Li, "Markerless kinect-based hand tracking for robot teleoperation," *International Journal of Advanced Robotic Systems*, vol. 9, no. 2, p. 36, 2012.
- [43] G.-L. Du, P. Zhang, L.-Y. Yang, and Y.-B. Su, "Robot teleoperation using a vision-based manipulation method," in *International Conference on Audio, Language and Image Processing*, 2010.

- [44] I. Almetwally and M. Malle, “Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera,” in *ICNSC*, 2013.
- [45] A. Sivakumar, K. Shaw, and D. Pathak, “Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube,” *arXiv preprint arXiv:2202.10448*, 2022.
- [46] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto, “Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation,” *arXiv preprint arXiv:2203.13251*, 2022.
- [47] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, “Sapien: A simulated part-based interactive environment,” in *CVPR*, 2020.
- [48] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols,” *arXiv*, 2015.
- [49] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “Mediapipe hands: On-device real-time hand tracking,” *arXiv preprint arXiv:2006.10214*, 2020.
- [50] Y. Rong, T. Shiratori, and H. Joo, “Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration,” *arXiv preprint arXiv:2008.08324*, 2020.
- [51] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, “Expressive body capture: 3d hands, face, and body from a single image,” in *CVPR*, 2019.
- [52] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe, “Collision detection: A survey,” in *International Conference on Systems, Man and Cybernetics*, 2007.
- [53] Nvidia, “PhysX physics engine,” <https://www.geforce.com/hardware/technology/physx>.
- [54] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, “Dexmv: Imitation learning for dexterous manipulation from human videos,” *arXiv preprint arXiv:2108.05877*, 2021.
- [55] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.
- [56] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *ICML*, 2015.
- [57] I.-M. Chen and J. W. Burdick, “Finding antipodal point grasps on irregularly shaped objects,” *IEEE transactions on Robotics and Automation*, vol. 9, no. 4, pp. 507–512, 1993.
- [58] UFactory, “Ufactory xarm6 robot,” <https://www.ufactory.cc/pages/xarm>.