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

Yuzhe Qin, Hao Su, and Xiaolong Wang

***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.**

***Index Terms*—Dexterous manipulation, imitation learning.**

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], [2]. However, the high Degree-of-Freedom (DoF) joints and discontinuous contact increase the *sample complexity* to train an RL policy. Besides, black-box optimization with RL rewards can also lead to *unexpected and unsafe* behaviors.

Manuscript received 24 February 2022; accepted 21 June 2022. Date of publication 3 August 2022; date of current version 23 August 2022. This letter was recommended for publication by Associate Editor T. Matsubara and Editor D. Kulic upon evaluation of the reviewers' comments. This work was supported in part by NSF under Grants CCF-2112665 through TILOS, 1730158 CI-New through CHASE-CI, and ACI-1541349 through CC*DNI Pacific Research Platform, and in part by Qualcomm and Meta. *(Hao Su and Xiaolong Wang contributed equally to this work.) (Corresponding author: Yuzhe Qin.)*

The authors are with the University of California San Diego, La Jolla, CA 92093 USA (e-mail: y1qin@eng.ucsd.edu; has168@eng.ucsd.edu; xiw012@ucsd.edu).

More videos can be found in https://yzqin.github.io/dex-teleop-imitation.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3196104, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3196104

Learning from human demonstration collected by teleoperation is a natural solution for dexterous manipulation given the similar morphology. Most current teleoperation systems require Virtual Reality (VR) [2]–[6] 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 and thus reduces the cost and is more scalable. However, vision-based teleoperation introduces new challenges. It needs to convert the human hand motion into robot motion command, which is called motion retargeting [7]–[9]. A human operator needs to choose the next-step movement based on the imagination of the future robot 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 letter, 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 iPhone 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 (Fig. 1, 3nd column in top 3 rows). The key of our system is to generate a customized robot hand on the fly in the physical simulator (Fig. 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 all the data collection, 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, as shown in last 3 columns in top 3 rows of Fig. 1). 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 [2].

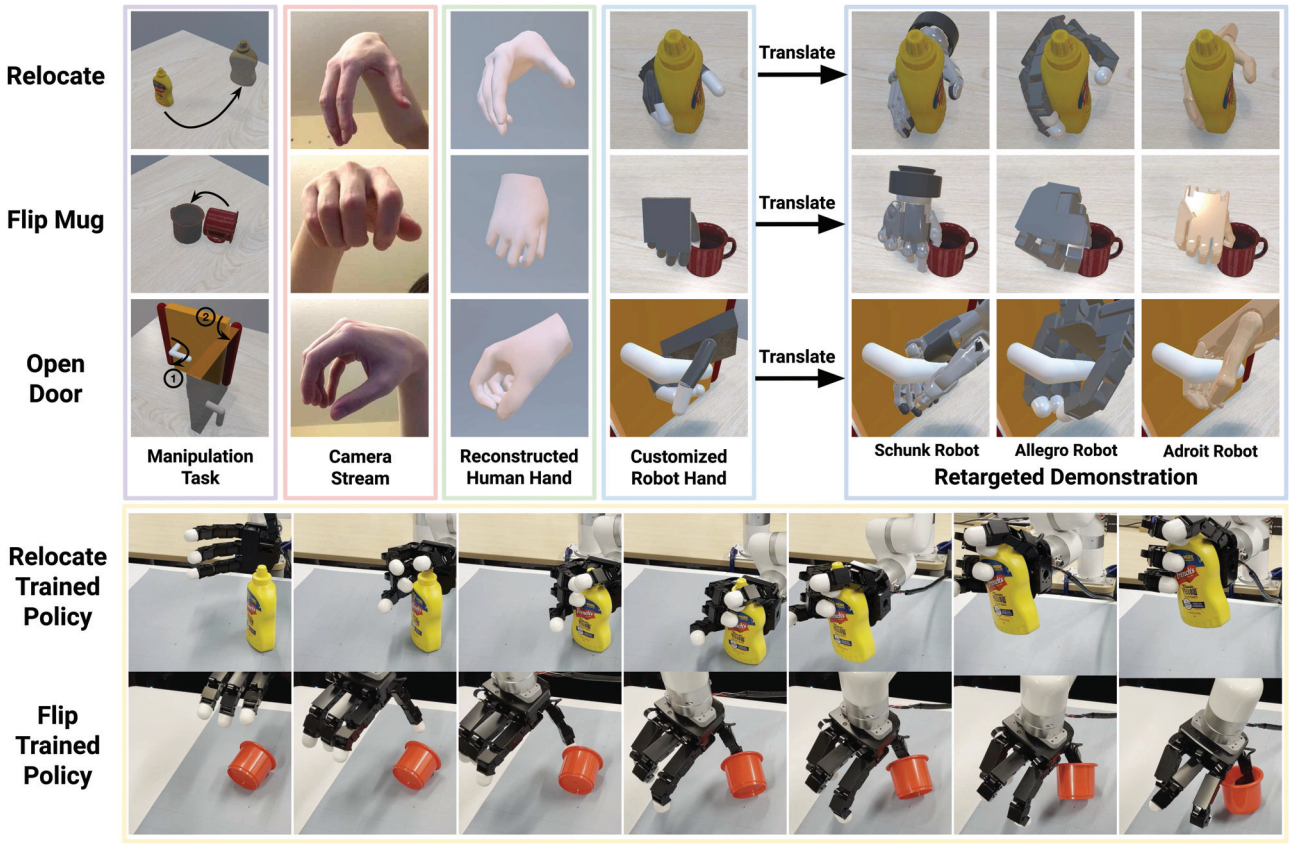


Fig. 1. We introduce a teleoperation system which utilizes a single camera on an iPad to stream a human hand, estimates the hand pose and shape, and converts it to a customized robot hand in a physical simulator for dexterous manipulation. Once the manipulation trajectories are collected, we translate 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.

We experiment with three types of challenging dexterous manipulation tasks: Relocate, Flip, and Open Door as shown in Fig. 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 an Allegro Hand attached on the XArm-6 robot in the real world (Fig. 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 [10]–[13]. Recently, we have witnessed Reinforcement Learning (RL) approaches [1] 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 [14], [15], imitation learning has been widely studied in the form of Inverse Reinforcement Learning [16]–[19] and incorporating expert demonstrations into the RL objectives [2], [20]–[22]. Our work is highly inspired by Rajeswaran et al. [2], 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.

Vision-based Teleoperation: Vision-based teleoperation frees the operator from wearing data capture devices commonly used

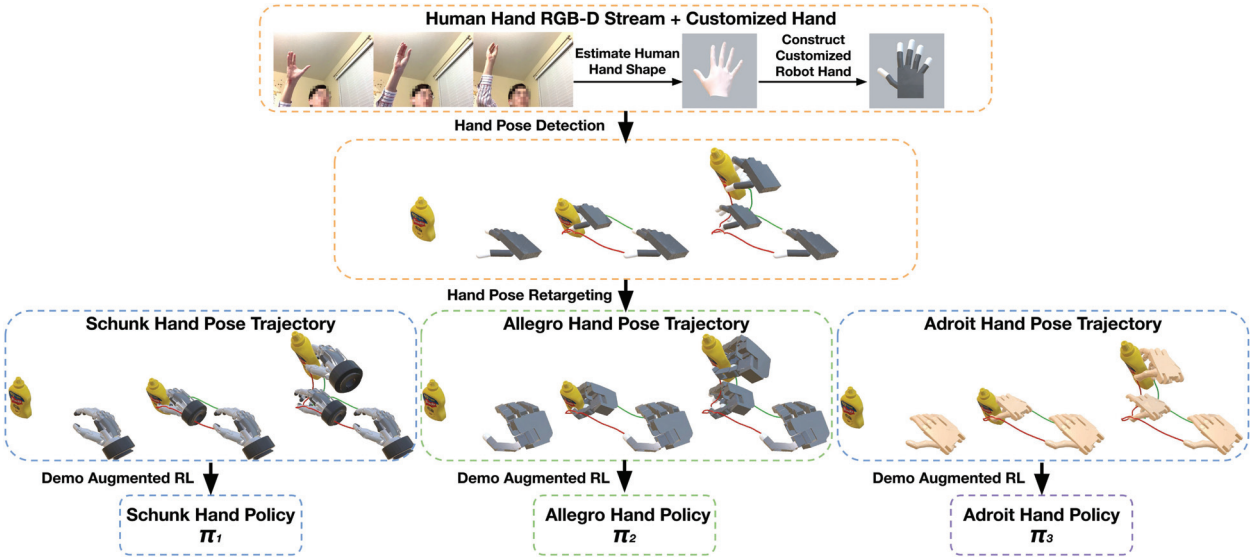


Fig. 2. **Overall Pipeline:** We stream the hand of human operator with an RGB-D camera. First we construct a customized robot hand in a physical simulator from estimated hand shape parameters result and teleoperate this robot to perform dexterous manipulation task. After teleoperation, we translate 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 second and third rows represent the finger tip trajectory. Different box color means different hand.

in game industry [23] and robot teleoperation [24], [24]–[26] on manipulation tasks, e.g. pick and place with a parallel gripper. DexPilot [7] is a pioneering work to extend the vision-based teleoperation to manipulation with an Allegro Hand[1]. To capture the hand pose, a black-clothed table with four calibrated RealSense cameras are used in their system. Our work only requires a single camera for teleoperation. Our novel customized robot hand provides a more intuitive way for data collection and allows generalization for learning with multiple robot hands, which has not been shown before.

III. Overview

We propose a novel framework in Fig. 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 available in the real-world, e.g., Allegro 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 robot hands 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 Fig. 1. In our experiments, we show learning with our demonstrations can significantly increase the robustness of our policy against the Sim2Real gap.

[1][Online]. Available: https://www.wonikrobotics.com/research-robot-hand

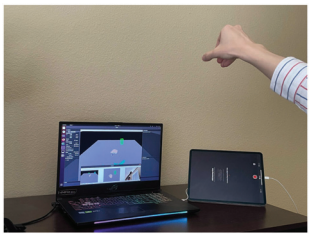


Fig. 3. Hardware setup with an iPad and a computer.

IV. Customized Hand Teleoperation

The hardware of our teleoperation system consists of an iPad and a laptop as shown in Fig. 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, and a GUI to visualize the current simulation environment.

A. Task Description

We construct our physical simulator based on SAPIEN [27], and design multiple dexterous manipulation tasks there.

Relocate. The robot picks up an object and moves it to a target position. It requires the agent to manipulate the object to match the given goal. The first row of Fig. 1 illustrates the relocate task. We experiment with three objects including *Tomato Soup Can*, *Potted Meat Can* and *Mustard Bottle*. The goal is visualized using the transparent object (as shown in Fig. 2). 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 Fig. 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 in a certain direction. We *randomize the position and the horizontal rotation* of the mug for each trial.

Open Door. As shown in the third row of Fig. 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 the RGB-D frames from camera as input. The outputs are composed of three parts: (i) wrist pose, which is the absolute position and orientation of hand wrist in the camera space; (ii) hand pose parameters, which is 15 rotations to specify the joint value of finger joints represented in axis-angle format; (iii) hand shape parameters, which is a 10-d vector represents the hand geometry at initial pose. It characterizes the shape of finger and palm in SMPL-X [28] representation. First, we use MediaPipe [29] 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 [30] model. Frankmocap takes cropped hand images as input and regresses the 10-d shape and 45-d pose parameters of the human hand. We use SMPL-X model to represent pose and shape parameters, where the shape parameters capture the hand geometry and the pose parameters capture the hand deformation using joint rotation. Given the shape and pose parameters, we can reconstruct a hand in the canonical frame where the wrist is placed at the origin. To compute wrist pose, we adopt the Perspective-n-Point (PnP) algorithm to match the key points in canonical frame and the detected key points in camera frame to solve the transformation from wrist to camera.

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. A desired property of the built customized robot hand is that we can directly apply the human hand pose parameters from hand detector without any motion retargeting. We extract the joint skeleton of the human hand (the red lines in Fig. 4) and create a robot model with the exactly the same kinematics tree. The relative transformation between each consequent joint pairs is preserved between human and robot hand. And the joint property is also kept the same. We choose primitive shapes, e.g. box for the palm and capsules for fingers, for efficient collision detection [31] and stable simulation [32]. 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. Fig. 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.

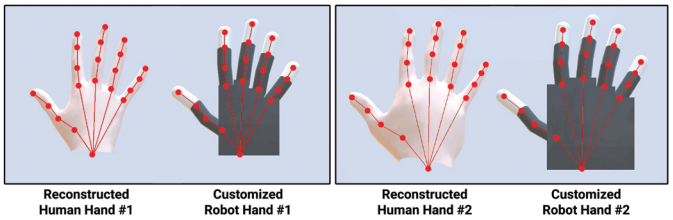


Fig. 4. Illustration of different customized robot hands generated from different human hands. The hand on left and right comes from different human. The red lines visualize the kinematics tree.

TABLE I
DoF Comparison for Different Robot Models. Customized Stands for the Customized Robot Hand

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)

D. System Efficiency

We use a laptop with an RTX 2070 GPU. The processing time for each RGB-D frame is less than 30 ms, which includes image-processing ($<$ 1ms), hand detection (3 ms), hand pose regression (25 ms), and control of the customized hand ($<$ 1ms). We choose to stream the images in 25 fps with size (640,480), but higher fps is also possible. The initialization process usually takes less than 5 s.

V. Multi-Robots Demonstration Translation

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 motion retargeting. With our customized hand, we can skip the computationally-heavy motion retargeting during teleportation and do it offline. We use optimization based motion retargeting to process the demonstration. The optimization objective is defined based on N link positions pairs define on both hands, e.g. thumb tip position,

specified by robot URDF file.

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

where q_t^C is joint angles at step t for customized robot and q_t^R is the counterpart for a specific robot, e.g. Schunk Robot Hand. We use f_i^C and f_i^R to represent the forward kinematics function of $i-th$ link position on the two robots. To improve the temporal consistency, we add a normalization term to penalize the joint angle change and initialize q_t^C from q_{t-1}^C. With retargeted kinematics motion, we can then estimate the action of robot controller following [33]: first smooth the joint trajectory with a low-pass filter, then compute the robot action using analytical inverse dynamics.

VI. Demonstration-Augmented Policy Learning

Given the retargeted demonstration, we perform imitation learning to solve the dexterous tasks defined in Section IV-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) [2] formulated below as our imitation algorithm.

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

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. Intuitively, the policy receive more information from demonstration during the initial stage of training while the policy learns more from the interaction trajectory afterwards.

VII. Experiment

We first demonstrate the benefits of using the proposed customized robot hand in teleoperation for data collection a user study. Then we show that the demonstrations collected by our system can improve the policy learning performance by a large margin on various tasks in simulated environment. Finally, we perform real-world experiments, 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.

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	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 III
Success Rate and Completion Time for *Open Door* 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**

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 Fig. 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

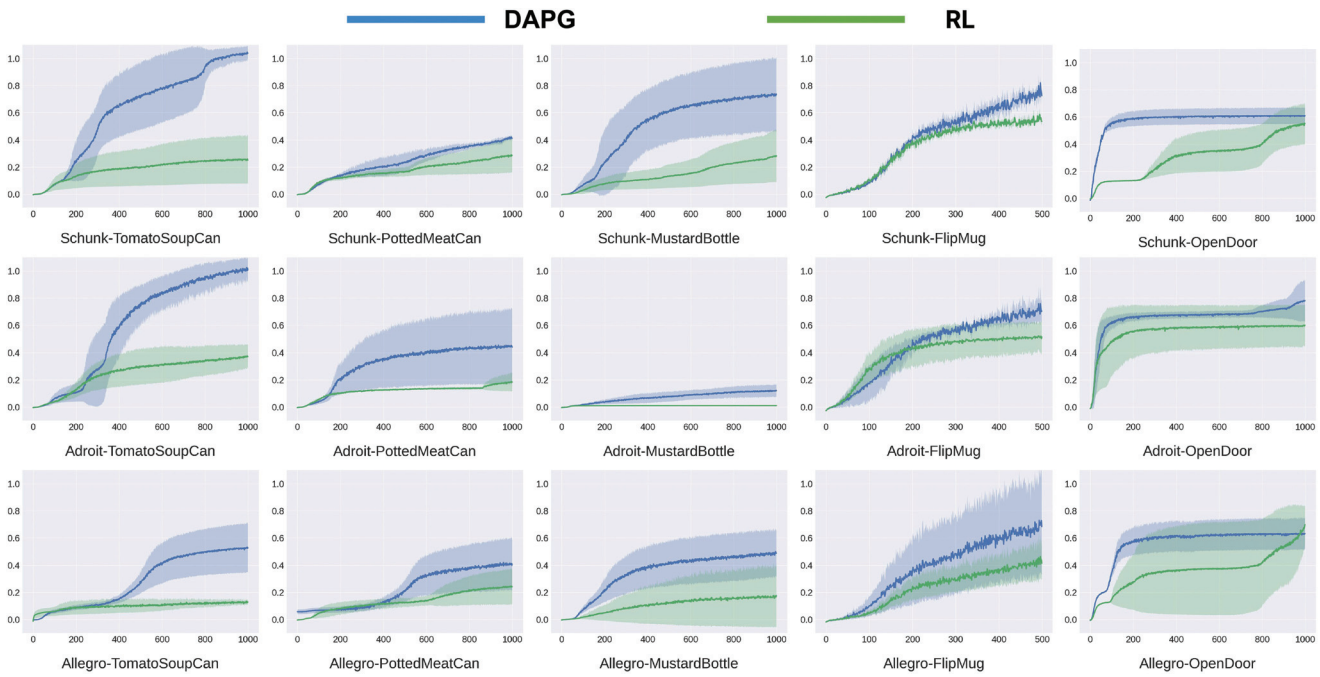


Fig. 5. Learning curves of RL and DAPG on all tasks with four different robot hands in four rows. The first three columns are Relocate task with three objects. Following two columns are Flip Mug and Open Door tasks. The x-axis is training iterations and y-axis is the normalized return. The shaded area indicates the standard deviation for three random seeds.

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 193 mm for adult males and is 172 mm for adult females.

B. Task Learning Comparison

Training. 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 use Trust Region Policy Optimization(TRPO) [34] 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 collect 50 trajectories of demonstration for each task and retarget the motion from customized hand to the specified robot. We train policies with three random seeds.

State and Action. 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 while the finger joints are actuated by position controllers.

We evaluate both RL and DAPG on *Relocate*, *Flip*, and *Open Door* tasks. The training curves are shown in Fig. 5. 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°. For *Open Door*, the task is successful when the joint angle of door hinge is larger than 60°.

As shown in Fig. 5 and Table IV, imitation learning method outperforms the RL baseline for most tasks and robots. It 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 Fig. 7: 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.

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. Fig. 6(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

TABLE IV
SUCCESS RATE OF THE EVALUATED METHODS. WE USE ± 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

Schunk Robot

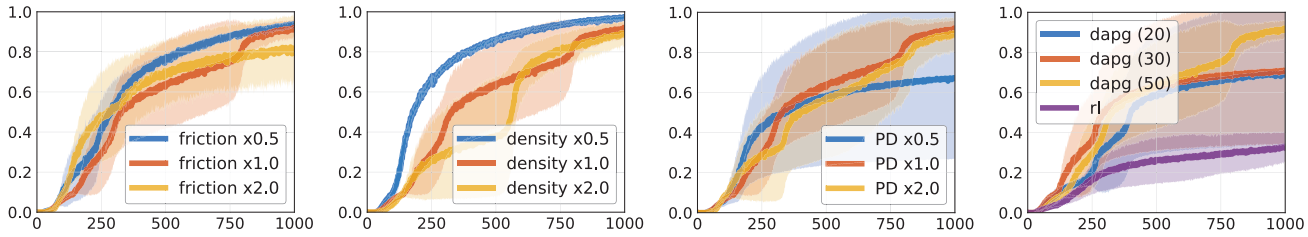
Task	RL	DAPG
Relocate-Toma.	45.3 ± 4.0	**85.0 ± 12.3**
Relocate-Pott.	6.7 ± 6.3	**41.0 ± 20.3**
Relocate-Must.	44.0 ± 20.0	**75.3 ± 24.7**
Flip-Mug	48.0 ± 4.3	**77.3 ± 5.0**
Open-Door	55.0 ± 14.3	**69.0 ± 7.7**

Adroit 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**

Allegro 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



(a) Object Friction (b) Object Density (c) PD Controller Params (d) Num of Demonstrations

Fig. 6. **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.

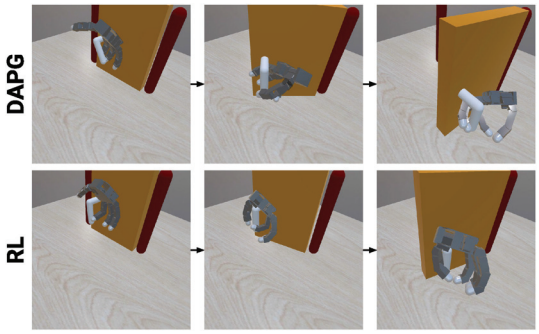


Fig. 7. Comparison of the naturalness on *Open Door* using Allegro Robot Hand. **Top Row**: policy learned by DAPG with demonstrations; **Bottom Row**: policy learned by RL without demonstrations.

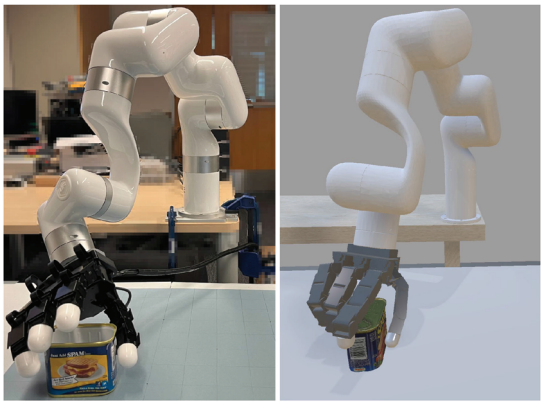


Fig. 8. **Left side:** real robot setup. The cyan poster on the table is a reference coordinate to determine whether the object is moved to the target position. **Right side:** simulated robot arm setup.

antipodal grasp [35], 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 Fig. 6(b). Fig. 6(c) illustrates the influence of controller parameters. Fig. 6(d) shows more demonstrations can achieve better performance.

D. Real-World Robot Experiments

In the real-world robot experiments, we attached an Allegro hand onto a XArm-6 robot arm[2] instead of using a flying hand. The experiment setup is shown in Fig. 8. We evaluate on the Relocate and Flip tasks. In simulation, we also build the same XArm6+Allegro model as real-world robot. To facilitate the sim2real transfer, we apply additive Gaussian noise onto the object pose to the observation and randomize the dynamics 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) with point cloud captured by a RealSense D435.

For *Relocate*, we randomize the initial and target object position for each evaluation trail. The initial object position is randomized within a 20 cm square and the target object position is randomized within a 40 cm square with fixed height. The

[2][Online]. Available: https://www.ufactory.cc/pages/xarm

Fig. 9. Objects Used in Experiments: Visualization of known objects and novel objects used in Relocate task. The first row shows the grasping process and the second row show the object. We test on three known objects and five novel objects.

TABLE V
SUCCESS RATE OF THE EVALUATED METHODS ON RELOCATE AND FLIP TASKS. WE USE ± TO REPRESENT MEAN AND STANDARD DEVIATION OVER THREE RANDOM SEEDS

Task	RL	DAPG
Relocate-Known.	22.2 ± 22.2	$\mathbf{77.7 \pm 11.1}$
Relocate-Novel.	18.5 ± 23.1	$\mathbf{66.6 \pm 11.1}$
Flip Mug	3.6 ± 6.4	$\mathbf{44.4 \pm 19.2}$

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 Fig. 8 to record the xy position of object. If the difference of of xy position between object and target is within 5 cm, 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 Fig. 9, 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 second bottom row of Fig. 1.

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

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 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.

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.

Limitations. Our pipeline is designed for dexterous hand. The performance will drop for *Open Door* if we use the retargeted demonstrations to train a parallel gripper since parallel gripper can not perform power grasp.

REFERENCES

[1] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2018.
[2] A. Rajeswaran et al., "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. Robot. Sci. Syst.*, 2018, p. 49.
[3] V. Kumar and E. Todorov, "MuJoCo HAPTIX: A virtual reality system for hand manipulation," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots*, 2015, pp. 657–663.
[4] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proc. ACM/IEEE 13th Int. Conf. Hum.-Robot Interaction*, 2018, pp. 78–86.
[5] 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," *Robot. Comput.-Integr. Manuf.*, vol. 71, 2021, Art. no. 102167.
[6] T. Zhang et al., "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5628–5635.
[7] A. Handa et al., "DexPilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9164–9170.
[8] S. Li et al., "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 416–422.
[9] D. Antotsiou, G. Garcia-Hernando, and T.-K. Kim, "Task-oriented hand motion retargeting for dexterous manipulation imitation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 287–301.
[10] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-D objects," *Int. J. Robot. Res.*, vol. 18, no. 4, pp. 355–381, 1999.

[11] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, pp. 255–262.
[12] S. Andrews and P. G. Kry, "Goal directed multi-finger manipulation: Control policies and analysis," *Comput. Graph.*, vol. 37, pp. 830–839, 2013.
[13] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1560–1565.
[14] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313.
[15] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," in *Proc. Conf. Robot Learn.*, 2020, pp. 1992–2005.
[16] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 1.
[17] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.
[18] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," 2017, *arXiv:1710.11248*.
[19] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," 2018, *arXiv:1807.06158*.
[20] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, 2008.
[21] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.
[22] I. Radosavovic, X. Wang, L. Pinto, and J. Malik, "State-only imitation learning for dexterous manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7865–7871.
[23] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.
[24] J. Kofman, S. Verma, and X. Wu, "Robot-manipulator teleoperation by markerless vision-based hand-arm tracking," *Int. J. Optomechatronics*, vol. 1, no. 3, pp. 331–357, 2007.
[25] G. Du, P. Zhang, J. Mai, and Z. Li, "Markerless kinect-based hand tracking for robot teleoperation," *Int. J. Adv. Robot.*, vol. 9, no. 2, pp. 36–45, 2012.
[26] G.-L. Du, P. Zhang, L.-Y. Yang, and Y.-B. Su, "Robot teleoperation using a vision-based manipulation method," in *Proc. Int. Conf. Audio Lang. Image Process.*, 2010, pp. 945–949.
[27] F. Xiang et al., "SAPIEN: A simulated part-based interactive environment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11094–11104.
[28] G. Pavlakos et al., "Expressive body capture: 3D hands, face, and body from a single image," *in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10975–10985.
[29] F. Zhang et al., "Mediapipe hands: On-device real-time hand tracking," 2020, *arXiv:2006.10214*.
[30] Y. Rong, T. Shiratori, and H. Joo, "Frankmocap: Fast monocular 3D hand and body motion capture by regression and integration," 2020, *arXiv:2008.08324*.
[31] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe, "Collision detection: A survey," in *Proc. Int. Conf. Syst. Man Cybern.*, 2007, pp. 4046–4051.
[32] Y. Rong, T. Shiratori, and H. Joo, "FrankMocap: Monocular 3D Whole-Body Pose Estimation System via Regression and Integration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, Montreal, BC, Canada, Nov. 17, 2021, pp. 1749–1759.
[33] Y. Qin et al., "DexMV: Imitation learning for dexterous manipulation from human videos," 2021, *arXiv:2108.05877*.
[34] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[35] I.-M. Chen and J. W. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *IEEE Trans. Robot. Autom.*, vol. 9, no. 4, pp. 507–512, Aug. 1993.