# Bridging the Human to Robot Dexterity Gap through Object-Oriented Rewards

Irmak Guzey[†] Yinlong Dai Georgy Savva Raunaq Bhirangi Lerrel Pinto

New York University

object-rewards.github.io

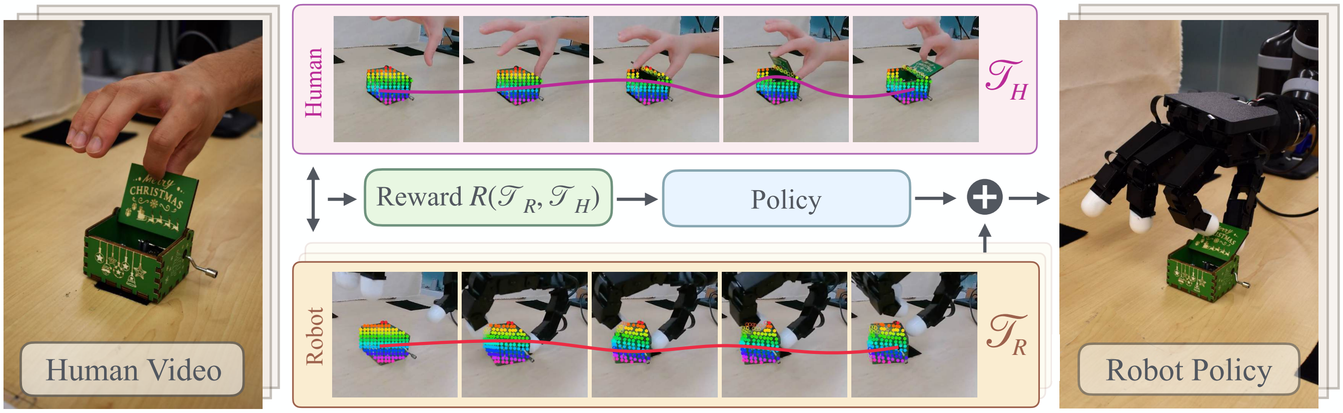

Fig. 1: HUDOR generates rewards from human videos by tracking points on the manipulable object, indicated by the rainbow-colored dots, over the trajectory. This allows for online training of multi-fingered robot hands given only a *single* video of a human solving the task (left) and without robot teleoperation. To optimize the robot's policy (middle), rewards are computed by matching the point movements of the robot policy $\mathcal{T}_R$ with those in the human video $\mathcal{T}_H$. In under an hour of online fine-tuning, our Allegro robot hand (right) is able to *open the music box*.

***Abstract*— Training robots directly from human videos is an emerging area in robotics and computer vision. While there has been notable progress with two-fingered grippers, learning autonomous tasks without teleoperation remains a difficult problem for multi-fingered robot hands. A key reason for this difficulty is that a policy trained on human hands may not directly transfer to a robot hand with a different morphology. In this work, we present HUDOR, a technique that enables online fine-tuning of the policy by constructing a reward function from the human video. Importantly, this reward function is built using object-oriented rewards derived from off-the-shelf point trackers, which allows for meaningful learning signals even when the robot hand is in the visual observation, while the human hand is used to construct the reward. Given a single video of human solving a task, such as gently opening a music box, HUDOR allows our four-fingered Allegro hand to learn this task with just an hour of online interaction. Our experiments across four tasks, show that HUDOR outperforms alternatives with an average of 4× improvement. Code and videos are available on our website https://object-rewards.github.io/.**

## I. Introduction

Humans effortlessly perform a wide range of dexterous tasks in their daily lives [1]. Achieving similar capabilities in robots is essential for their effective deployment in the real world. Towards this end, recent advances have enabled the learning of multi-modal, long-horizon, and dexterous behaviors for two-fingered grippers [2, 3, 4, 5] using imitation learning (IL) from teleoperated robot data. However, extending such methods to more complex tasks with multi-fingered hands has proven challenging.

The difficulty with using teleoperation-based learning for multi-fingered hands stems from two key issues. First, this requires large amounts of data for moderate amounts of robustness. For example, even tasks with two-fingered grippers [6, 7, 8] often require thousands of demonstrations to train robust policies. This data requirement is likely greater for tasks demanding higher precision and dexterity. Second, teleoperating multi-fingered hands is a challenging systems problem due to the demand for lower-latency and continuous feedback when controlling multiple degrees of freedom [9, 10, 11].

An alternate approach that circumvents teleoperation is to develop policies for robots using first-person videos of humans executing tasks [12, 13, 14]. However, most previous approaches have required either additional teleoperated robot demonstrations [15] or human-intervened learning [16] for fine-tuning. This extra information is often necessary to bridge the gap between the morphological and visual differences between human hands (as seen in human video data) and robot hands (as observed in robot interactions).

[†] Correspondence to irmakguzey@nyu.edu.

In this work, we present HUDOR, a new approach to bridge the gap between human videos and robot policies through online imitation learning. Given a human video and hand pose trajectories, an initial robot replay can be generated using pose transformation and full robot inverse kinematics. However, this initial replay often fails due to morphological differences between human and robot hands. HUDOR improves this initial replay through following steps: (a) We track the points of the manipulated object in both human and robot trajectory videos; (b) then calculate the similarity of object motion and articulation using these tracked point sets, (c) and finally fine-tune the initial robot trajectory through inverse reinforcement learning. By iteratively refining its performance based on these comparisons, the robot effectively imitates human actions while adapting to its own physical constraints.

We evaluate HUDOR on four dexterous tasks, including opening a small music box with one hand and sliding and picking up a thin card. Our contributions can be summarized as follows:

1) HUDOR introduces the first framework that enables the learning of dexterous policies on multi-fingered robot hands using only a single human video and hand pose trajectory (Section IV).
2) HUDOR introduces a new object-oriented reward calculation method that matches human and robot trajectories. This approach gives 2.1× better performance on three of our tasks than common reward functions (Section IV-C).
3) HUDOR outperforms state-of-the-art offline imitation learning methods for learning from human demonstrations [16, 3], achieving an average improvement of 2.64×, emphasizing the need for online corrections (Section IV-B).

Robot videos are best viewed on our website: https://object-rewards.github.io/.

## II. Related Works

Our work draws inspiration from extensive research in dexterous manipulation, learning from human videos, and imitation learning. In this section, we focus our discussion on the most pertinent contributions across these interrelated areas.

*a) Robot Learning for Dexterous Manipulation:* Learning dexterous policies for multi-fingered hands has been a long-standing challenge that has captured the interest of the robotics learning community [17, 18, 9]. Some works have addressed this problem by training policies in simulation and deploying them in the real world [19, 18]. Although this approach has produced impressive results for in-hand manipulation [20, 21], closing the sim-to-real gap becomes cumbersome when manipulating in-scene objects.

Other works have focused on developing different teleoperation frameworks [22, 23, 10, 24] and training offline policies using robot data collected through these frameworks. While these frameworks are quite responsive, teleoperating dexterous hands without directly interacting with objects remains difficult for users due to the morphological mismatch between current robotic hands and the lack of tactile feedback for the teleoperator.

Given the challenges of large-scale data collection, most offline dexterous policies tend to fail due to overfitting. To mitigate this, some previous works have focused on learning policies with limited data [25, 26], either by using nearest-neighbor matching for action retrieval [25, 10, 22] or by initializing with a single demonstration and learning a residual policy with online interactions to improve generalization [26, 27].

*b) Learning from Human Videos:* With the goal of scaling up data collection using more accessible sources, the vision and robotics communities have worked on learning meaningful behaviors and patterns from human videos [28, 29, 30, 31]. Some efforts focus on learning simulators that closely mimic the real-world environment of the robot from human videos using generative models [29, 32, 28], using these simulators to train policies and make decisions by predicting potential future outcomes.

Other works use internet-scale human videos to learn higher-level skills or affordances [30, 33]. However, these works either require low-level policies to learn action primitives for interacting with objects [30], or only focus on simple tasks where a single point of contact is sufficient for manipulation [33]. Yet other approaches leverage on-scene human videos to learn multi-stage planning [15, 13], but need additional robot data to learn lower-level object interactions. Notably, all of these works focused on two-gripper robots, where manipulation capabilities are limited and objects are less articulated.

A recent study, DexCap [16], addresses this issue for dexterous hands by using multiple cameras and a hand motion capture system to collect human demonstrations. An offline policy is learned by masking the human hand from the environment point cloud followed by an online fine-tuning stage with human feedback. HUDOR differs from this work by eliminating the need for cumbersome human feedback by automatically extracting a reward from a single human demonstration and allowing the robot to learn from its mistakes to compensate for the morphology mismatch between the robot and the human.

## III. Learning Teleoperation-Free Online Dexterious Policies

HUDOR introduces a framework to learn dexterous policies from a single in-scene human video of task execution. Our method involves three steps: (1) A human video and corresponding hand pose trajectory are recorded; (2) hand poses are transferred and executed on the robot using pose transformation and full-robot inverse kinematics (IK); and (3) inverse reinforcement learning (IRL) is used to successfully imitate the expert trajectory. In this section, we explain each component in detail.

### A. Robot Setup and Human Data Collection

Our hardware setup includes a Kinova JACO 6-DOF robotic arm with a 16-DOF four-fingered Allegro hand [9]

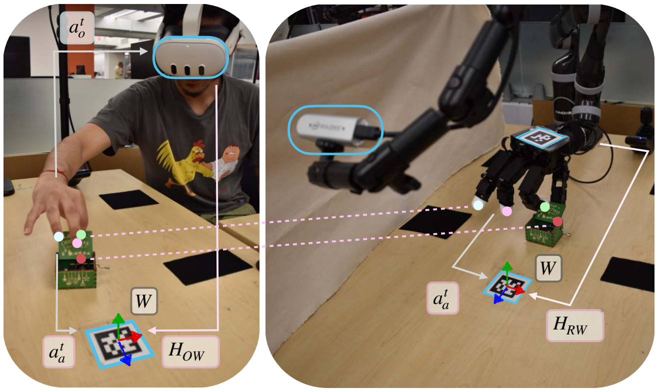

Fig. 2: Illustration of the robot setup and trajectory transfer in HUDOR. Aruco markers are used for calibration. Here, the VR headset is used solely for obtaining hand pose estimates and can be worn or attached to the setup as needed. World frame $W$ is visualized on the ArUco marker on the operation table.

attached. Two RealSense RGBD cameras [34] are positioned around the operation table for calibration and visual data collection. A Meta Quest 3 VR headset is used to collect hand pose estimates. Our first step involves computing the relative transformation between the Quest frame and the robot frame to directly transfer the recorded hand pose trajectory from the human video to the robot as shown in Figure 2. We use two ArUco markers – one on the operation table and another on top of the Allegro hand – to compute relative transformations between camera frames. The first marker is used to define a world frame and transform fingertip positions from the Quest frame to the world frame, while the second marker is used to determine the transformation between the robot's base and the world frame. IG: add how we initialize the wrist

*a) Relative Transformations:* We collect human hand pose estimates using existing hand pose detectors on Quest 3, and capture visual data using the RGBD cameras. Fingertip pose for the $i^{\text{th}}$ human fingertip captured in the headset frame at time $t$, $a_o^{t,i}$, are first transformed to the world frame as $a_w^{t,i} = H_{OW} \times a_o^{t,i}$, where $H_{OW}$ is the homogeneous transform from the headset frame to the world frame. This transform is computed by detecting the ArUco marker on the table using the cameras on the VR headset. A standard calibration procedure [35] is used to compute the transformation $H_{RW}$ between the robot frame and the world frame by detecting the two ArUco markers using the RGB camera shown in Figure 2. This calibration allows us to directly transfer human fingertip positions from the Oculus headset to the robot's base using the equation:

$$\begin{aligned} a_r^{t,i} &= H_{RW}^{-1} \times a_w^{t,i} \qquad (1) \\ &= H_{RW}^{-1} \times H_{OW} \times a_o^{t,i} \qquad (2) \end{aligned}$$

where, $a_r^{t,i}$ are the homogeneous coordinates of the $i^{\text{th}}$ human fingertip positions from the recorded video in the robot frame. Henceforth, we use $a_r^t = [a_r^{t,0}, a_r^{t,1}, a_r^{t,2} a_r^{t,3}]$ to refer to the 12-dimensional vector containing concatenated locations of the four fingertips in robot frame.

*b) Data alignment:* During data collection, we record the fingertip positions $a_r^t$ and image data $o^t$ for all $t = 1 \ldots T$ where $T$ is the trajectory length. Since these components are collected at different frequencies, we align them on collected timestamps to produce synchronized tuples $(a_r^t, o^t)$ for each time $t$. The data is then subsampled to 5 Hz.

*c) Inverse Kinematics:* To ensure the robot's fingertips follow the desired positions relative to its base, we implemented a custom inverse kinematics (IK) module for the full robot arm-hand system. This module uses gradient descent on the joint positions, using the Jacobians of the robot with respect to the desired fingertip pose changes [36]. We apply different learning rates for the hand and arm joints, allowing the IK to prioritize the hand movements. The hand learning rate is set to be 50 times higher than the arm learning rate, enabling more natural and precise control of the fingers. The IK module takes the desired fingertip positions $a_r^t$ and the current joint positions of both hand and arm $j^t$ as inputs, and outputs the next joint positions $j^{*t+1} = I(a_r^t, j^t)$ needed to reach the target.

Using the calibration and IK procedures outlined above, our robot arm-hand system can follow a fingertip trajectory directly from an in-scene human video.

### B. Residual Policy Learning

Due to the morphological differences between the human and robot, as well as errors in VR hand pose estimation, naively replaying the retargeted fingertip trajectories on the robot mostly does not successfully solve the task, even when the object is in the same location. To alleviate this problem, we learn an online residual policy using inverse reinforcement learning (IRL) to augment the trajectory replay. Traditional IRL algorithms rely on reward functions derived from straightforward methods such as image-based matching rewards [27, 26] using in-domain demonstration data. However, due to the significant difference in visual appearance of human and robot hands, these methods do not provide effective reward signals. To get around this domain gap, we propose a novel algorithm for object-centric trajectory-matching rewards.

*a) Object Point Tracking and Trajectory Matching:* Our reward computation involves using off-the-shelf computer vision models to track the motion of points on the object of interest. We compute the mean squared error between motion of these points in the human expert video and the robot policy rollout and use this as a reward at every timestep in our online learning framework. In this section, we explain our reward calculation in detail.

- *Object State Extraction*: Given a trajectory $\tau = [o^1, \ldots, o^T]$, where $T$ is the length of the trajectory and $o^t$ is an RGB image at time $t$, we use the first frame $o^1$ as input to a language-grounded Segment-Anything Model [37, 38] – `langSAM`. `langSAM` uses a text prompt and GroundingDINO [39] to extract bounding boxes for the object, which are then input to the SAM [38] to generate a mask. The output of `langSAM` corresponding to $o^1$ is a segmentation mask

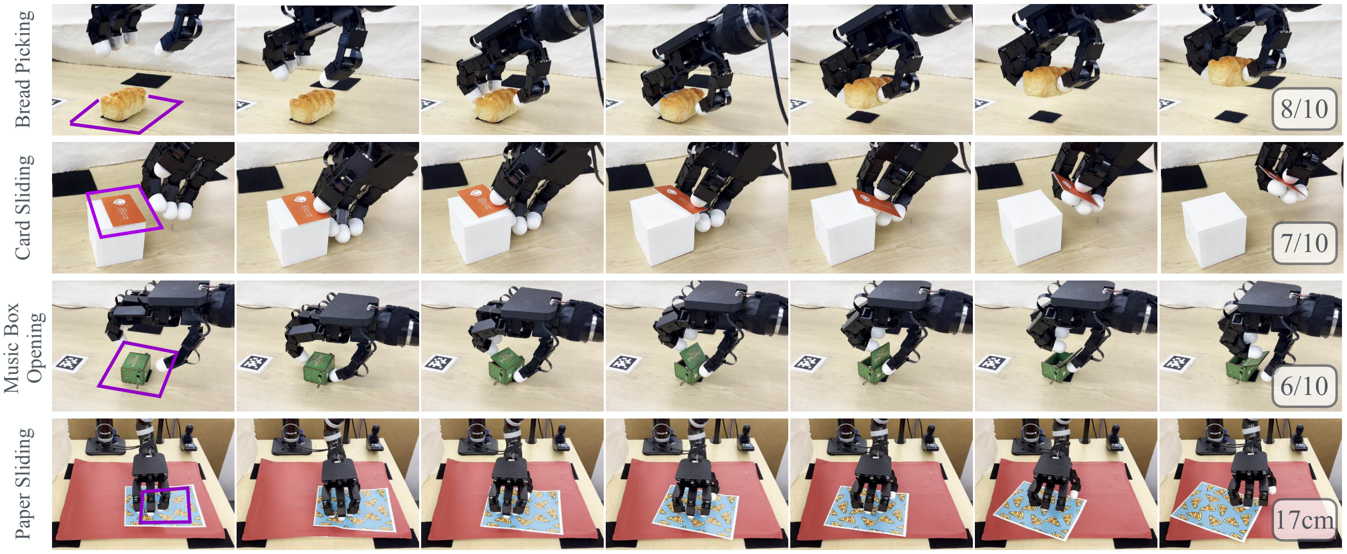

Fig. 3: Rollouts of trained policies from HuDOR on four tasks are shown. For all tasks, validation is performed at various locations within the illustrated areas in the leftmost frames, while training is conducted using a single human video where the initial object configuration is in the middle of these areas. Success for each task is shown in the rightmost frames. Videos are best viewed on our website: https://object-rewards.github.io/.

for the initial object position, $P^1 \in \mathbb{R}^{N\times 2}$, which is represented as a set of $N$ points on the object, where $N$ is a hyperparameter. The parameter $N$ determines the density of object tracking and is adjusted based on the object's size in the camera view.

- *Point Tracking*: The mask $P^1$ is used to initialize the transformer-based point tracking algorithm Co-Tracker [40]. Given a trajectory of RGB images, $\tau$, and the first-frame segmentation mask, $P^1$, Co-Tracker tracks points $p_i^t = (x_i^t, y_i^t)$ in the image throughout the trajectory $\tau$ for all $t \in \{1 \dots T\}$, where $P^1 = [p_1^1, \dots p_N^1]$. We use $\tau^p = [P^1, \dots P^T]$ to denote the point trajectory consisting of the sets of tracked points.
- *Matching the Trajectories*: First, we define two additional quantities: mean translation at time $t$, $\delta^t_{trans}$, and mean rotation at time $t$, $\delta^t_{rot}$. $\delta^t_{trans}$ is defined as the mean displacement of all the points in $P^t$ from $P^1$. Similarly, $\delta^t_{rot}$ is defined as the mean rotation vector about the centroid of all the points in $P^t$ from $P^1$. Concretely,

$$\delta^t_{trans} = \mathbb{E}_i(p_i^t - p_i^1) \tag{3}$$

$$\delta^t_{rot} = \mathbb{E}_i\left[\left(p_i^t - \mathbb{E}_i(p_i^t)\right) \times \left(p_i^1 - \mathbb{E}_i(p_i^1)\right)\right] \tag{4}$$

We define the *object motion* at time $t$ as $\mathcal{T}^t = [\delta^t_{trans}, \delta^t_{rot}]$. Given two separate point trajectories, one corresponding to the robot $\tau^p_R$ and one corresponding to the human $\tau^p_H$, the reward at time $t$ is calculated by computing the root mean squared error between the object motions of the robot and human at time $t$:

$$r_t^{H2R} = -\sqrt{\left(|\mathcal{T}_R^t - \mathcal{T}_H^t|^2\right)} \tag{5}$$

*b) Exploration Strategy:* We select a subset of action dimensions to explore and learn from. This speeds up the learning process and enables quick adaptation. For exploration, we use a scheduled additive Ornstein-Uhlenbeck (OU) noise [41, 42] to ensure smooth robot actions.

After extracting a meaningful reward function and identifying a relevant subset of the action space, we learn a residual policy $\pi_r(\cdot)$ on this subset by maximizing the following reward function for each episode $i$ using DrQv2 [43]:

$$R_i^{H2R} = \sum_t r_t^{H2R} \tag{6}$$

Inputs to the residual policy $a_+^t = \pi_r(a_r^t, \Delta s^t, \mathbb{E}(P_R^t), \mathcal{T}_R^t)$ at time $t$ are: (a) the human retargeted fingertip positions with respect to the robot's base $a_r^t$, (b) change in current robot fingertip positions $\Delta s^t = s^t - s^{t-1}$, (c) the centroid of the tracked points set $\mathbb{E}(P_R^t)$ and (d) the object motion at time $t$, $\mathcal{T}_R^t$. Finally, we compute the executed actions as follows:

$$\begin{aligned} a^t &= a_r^t + a_+^t & (7)\\ &= a_r^t + \pi_r(a_r^t, \Delta s^t, \mathbb{E}(P_R^t), \mathcal{T}_R^t) & (8) \end{aligned}$$

The action, $a^t$, is sent to the IK module which converts it into joint commands for the robot. The policy is improved over time using DrQv2 as the robot accumulates experience interacting with the object.

## IV. Experimental Evaluation

We evaluate our method against 6 different baselines and run multiple ablations to answer the following questions:

1) How much do online corrections improve the performance of HuDOR?
2) Does HuDOR improve over common reward functions?
3) How well does HuDOR generalize to new objects?

### A. Task Descriptions

We experiment with four dexterous tasks, which are visualized in Figure 3. Exploration axes mentioned are with respect to the base of the robot.

*a) **Bread Picking**:* The robot must locate an orange-colored piece of bread, pick it up, and hold it steadily for a sustained period. During validation, the bread moves and rotates within a 15cm × 10cm space. We explore on X axes of all fingers for this task. Text prompt used to retrieve the mask is *orange bread*.

*b) **Card Sliding**:* The robot must locate and slide a thin card with its thumb and pick it up by supporting it with the rest of its fingers. During validation, the card rotates and moves within a 10cm × 10cm space during validation. Text prompt used to retrieve the object mask is *orange card*. We explore on only X and Y axes of the thumb.

*c) **Music Box Opening**:* The robot must locate and open a small music box. It uses its thumb to stabilize the box while unlatching the top with its index finger. During validation, the box moves and rotates within a 10cm × 10cm space. We explore on all axes of thumb and index fingers. Text prompt used to retrieve the mask is *green music box*. For this task specifically we use sparse rewards and use only the last 5 frames of the trajectory for reward calculation for HUDOR and all of our baselines.

*d) **Paper Sliding**:* The robot must slide a given piece of paper to the right. During validation, the paper rotates and moves within a 15cm × 15cm space. Text prompt used to retrieve the mask is *blue paper with pizza patterns*. Higher rewards are given as the paper moves further to the right. Success in this task is measured by the distance the paper moves to the right, expressed in centimeters. We explore on X and Z axes of all the fingers.

*Evaluating robot performance*: To compare the robot's performance, we evaluate HUDOR against various online and offline algorithms. For all online algorithms, we train the policies until the reward converges, up to one hour of online interactions. We evaluate the methods by running rollouts on 10 varying initial object configurations for every task.

### B. How important are online corrections?

Figure 4 demonstrates how online learning improves the policy in the Paper Sliding task. As can be seen, HUDOR enables the robot policy trajectory to move progressively closer to that of the human expert. To showcase the importance of online corrections, we implement and run the state-of-the-art transformer-based behavior cloning (BC) algorithm VQ-Bet [3], as the base architecture for all of our offline baselines. We ablate the input and the amount of demonstrations used to experiment on different aspects, and compare HUDOR against the following offline baselines:

1) **BC - 1 Demo** [3]: HUDOR enables training robust policies with only a single human demonstration. For fairness, we compare its offline counterpart and train VQ-Bet end-to-end using only a single demonstration per task. The centroid of the tracked points set, the rotation and translation of the object, and the robot's fingertip positions are given as input to the model.

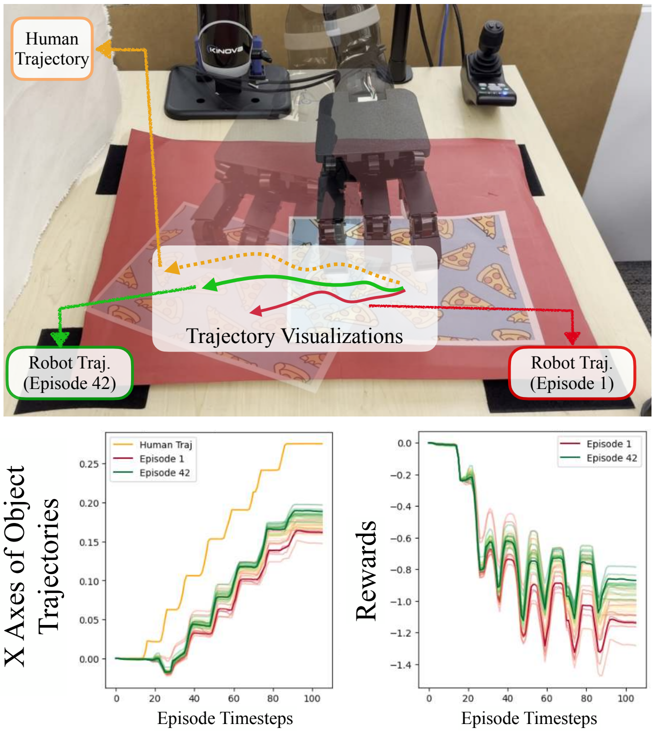

Fig. 4: Illustration of how online correction improves robot policy and moves the robot trajectory closer to the expert's as time progresses in the Paper Sliding task. At the top, we visualize the trajectories of the paper in different episodes and in the human video. At the bottom, we showcase the X-axis of the trajectory of the tracked points for different episodes and their corresponding rewards. The color of the episodes gradually changes from red in Episode 1 to green in Episode 42.

2) **BC** [3]: We run VQ-Bet similar to the previous baseline, but we use 30 demonstrations for each task.
3) **Point Cloud BC**: Similar to DexCap [16], we include point cloud in our input space and modify the input of the BC algorithm by concatenating the point cloud representations received from PointNet [44] encoder. We uniformly sample 5000 points from the point cloud and pass them to PointNet without any further preprocessing. Gradients are backpropagated through the entire system, including the point cloud encoder.

TABLE I: Comparison of HUDOR to different offline algorithms. Paper sliding success is measured in cm of rightward motion; other tasks show success rates out of 10 robot rollouts.

| Method | Bread (./10) | Card (./10) | Music (./10) | Paper (cm) |
|---|---|---|---|---|
| BC - 1 Demo [3] | 0 | 0 | 0 | 3.5 ± 1.1 |
| BC [3] | 3 | 0 | 0 | 4.1 ± 1.3 |
| Point Cloud BC [16] | 3 | 0 | 0 | 12 ± 1.3 |
| HUDOR (ours) | **8** | **7** | **6** | **17.3 ± 1.5** |

Table I shows the comparison results. As expected, the BC-1 Demo baseline quickly overfits and fails across all tasks. The BC baseline performs relatively well on the Bread Picking task, where precision is less critical. However, on tasks like Card Sliding and Music Box Opening, which require more dexterity, it also overfits quickly—reaching the objects but failing to maintain the necessary consistency. Both BC baselines manage to reach the paper but do little beyond that for Paper Sliding. Our strongest baseline, Point Cloud BC, performs relatively well on Paper Sliding and Bread Picking. We observed that with this baseline, when the robot hand occupies too much of the scene, the data goes out of distribution, causing the model to fail. It moves the paper to the middle but doesn't move it further, grasps the bread but fails to lift it, and reaches for the lid of the music box but struggles to stabilize it with the thumb. More of the failure cases can be seen on our website.

These results indicate that the dexterous skills learned using this human data collected with HuDOR can scale better with more data for some tasks. However, for highly precise tasks such as Music Box Opening and Card Sliding, all offline methods fail, highlighting the importance of online corrections for tasks requiring high dexterity.

### C. *Does* HuDOR *improve over other reward functions?*

In HuDOR in order to compensate for the visual differences between the human and robot videos, we use object-oriented point tracking based reward functions to guide the online learning. We ablate over our design decision, and train online policies for three of our tasks with the following reward functions:

1) **Image OT** [27]: We pass RGB images from both the robot and the human videos through pretrained Resnet-18 [45] image encoders to get image representations and apply optimal transport (OT) based matching on them to get the reward, similar to FISH [27].
2) **Pred OT**: Instead of direct images we apply OT matching on the points that are tracked throughout both the trajectories of tracked sets of points $\tau_r^p$ and $\tau_h^p$. IG: Change this PRED to Point

TABLE II: Comparison of success of HuDOR to different reward extraction algorithms. Success is shown as similar to Table I

| Method | HuDOR(ours) | Image OT [27] | Pred OT |
|---|---|---|---|
| Bread Picking | **8** | 6 | 6 |
| Music Box Opening | **6** | 1 | 2 |
| Paper Sliding (cm) | **17.25 ± 1.47** | 16.1 ± 1.37 | 16.5 ± 1.23 |

We present the success rates in Table II. We observe that in tasks where the object occupies a large area in the image and the visual differences between the hand and the robot do not significantly affect the image, the Image OT baseline performs similarly to HuDOR, as seen in the Paper Sliding task. However, in tasks where the camera needs to be closer to the object to detect its trajectory—such as Music Box Opening—the visual differences between the hand and the robot significantly hinder training, causing image-based reward calculations to fail. On the other hand, direct matching on the predicted points fails because the points tracked for two separate trajectories do not correspond to each other; the same indexed point in one trajectory may correspond to a different location on the object in another trajectory. These differences cause matching to give inconsistent reward emphasizing the importance of matching *trajectories* rather than points or images.

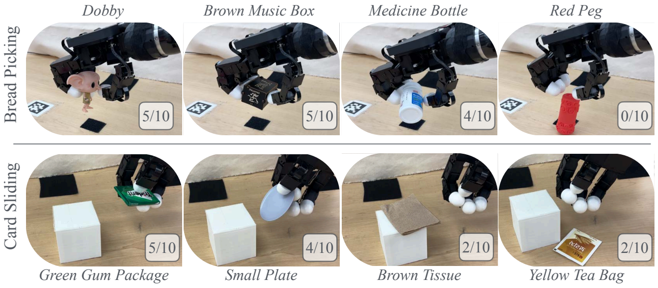

Fig. 5: Generalization experiments on Bread Picking and Card Sliding task. We input the text prompts on top and bottom as input to the language grounded SAM model to get the initial mask for each object.

### D. *How well does* HuDOR *generalize to new objects?*

We test the generalization capabilities of policies trained with HuDOR on new objects for two of our tasks, with the results shown in Figure 5. In these experiments, we directly run the policies on new objects without retraining, using different text prompts for each inference to obtain object segmentation. We observe that HuDOR can generalize with varying degrees of success to new objects when their shape and texture are not significantly different from the original object. While the weight and thickness of the card affect success in Card Sliding, texture is the most critical factor causing failure in Bread Picking. Despite the Dobby sculpture having a very different shape from the bread, HuDOR performs well; however, when the object is slippery, like the Red Peg, we observe complete failure. Most failures in Card Sliding occur when objects are so light that after sliding, they don't fall onto the supporting fingers, preventing a tight grip. These experiments show how point-tracking can enable some policy generalization.

## V. Limitations and Discussion

In this paper, we introduced HuDOR, a point-tracking, object-oriented reward mechanism designed to close the gap in human-to-robot policy transfer for dexterous hands. HuDOR improves upon both offline methods and online counterparts with different reward functions. We also demonstrate HuDOR's generalization to entirely new objects.

Despite its strengths, we identify three limitations. First, our framework only works with in-scene human videos. We believe integrating in-the-wild data collection would significantly enhance its generalization potential. Second, the exploration mechanism requires prior knowledge of which subset of action dimensions is suitable for exploration. Finally, there is no retry mechanism during an episode; when the robot makes a mistake, it can only retry in the next episode. This makes training for long-term tasks challenging. Incorporating a multi-stage learning framework could address this issue. These represent interesting opportunities for future improvements to HuDOR.

# References

[1] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383, 2016.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," *arXiv e-prints arXiv:2304.13705*, Apr. 2023.

[3] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. Mahi Shafiullah, and L. Pinto, "Behavior Generation with Latent Actions," *arXiv e-prints arXiv:2403.03181*, Mar. 2024.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024.

[5] Z. J. Cui, Y. Wang, N. Muhammad, L. Pinto, *et al.*, "From play to policy: Conditional behavior generation from uncurated robot data," *arXiv preprint arXiv:2210.10047*, 2022.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," *arXiv e-prints arXiv:2307.15818*, p. arXiv:2307.15818, July 2023.

[7] Open X-Embodiment Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," *arXiv e-prints arXiv:2310.08864*, Oct. 2023.

[8] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. Mahi Shafiullah, "Robot Utility Models: General Policies for Zero-Shot Deployment in New Environments," *arXiv e-prints arXiv:2409.05865*, Sept. 2024.

[9] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto, "Holodex: Teaching dexterity with immersive mixed reality," 2022.

[10] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto, "OPEN TEACH: A Versatile Teleoperation System for Robotic Manipulation," *arXiv e-prints arXiv:2403.07870*, Mar. 2024.

[11] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang, "Bunny-VisionPro: Real-Time Bimanual Dexterous Teleoperation for Imitation Learning," *arXiv e-prints arXiv:2407.03162*, July 2024.

[12] S. Kumar, J. Zamora, N. Hansen, R. Jangir, and X. Wang, "Graph Inverse Reinforcement Learning from Diverse Videos," *arXiv e-prints arXiv:2207.14299*, July 2022.

[13] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos," *arXiv e-prints arXiv:1912.04443*, Dec. 2019.

[14] C. Eze and C. Crick, "Learning by Watching: A Review of Video-based Learning Approaches for Robot Manipulation," *arXiv e-prints arXiv:2402.07127*, Feb. 2024.

[15] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar, "MimicPlay: Long-Horizon Imitation Learning by Watching Human Play," *arXiv e-prints arXiv:2302.12422*, Feb. 2023.

[16] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "DexCap: Scalable and Portable Mocap Data Collection System for Dexterous Manipulation," *arXiv e-prints arXiv:2403.07788*, Mar. 2024.

[17] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State, "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," *arXiv e-prints arXiv:2210.13702*, Oct. 2022.

[18] 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 e-prints arXiv:1910.07113*, Oct. 2019.

[19] K. Shaw, A. Agarwal, and D. Pathak, "LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning," *arXiv e-prints arXiv:2309.06440*, Sept. 2023.

[20] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without Seeing: Towards In-hand Dexterity through Touch," *arXiv e-prints arXiv:2303.10880*, Mar. 2023.

[21] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-Level Reward Design via Coding Large Language Models," *arXiv e-prints arXiv:2310.12931*, Oct. 2023.

[22] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto, "Holodex: Teaching dexterity with immersive mixed reality," *arXiv preprint arXiv:2210.06463*, 2022.

[23] S. Yang, M. Liu, Y. Qin, R. Ding, J. Li, X. Cheng, R. Yang, S. Yi, and X. Wang, "ACE: A Cross-Platform Visual-Exoskeletons System for Low-Cost Dexterous Teleoperation," *arXiv e-prints arXiv:2408.11805*, Aug. 2024.

[24] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9164–9170.

[25] I. Guzey, B. Evans, S. Chintala, and L. Pinto, "Dexterity from touch: Self-supervised pre-training of tactile representations with robotic play," 2023.

[26] I. Guzey, Y. Dai, B. Evans, S. Chintala, and L. Pinto, "See to Touch: Learning Tactile Dexterity through Visual Incentives," *arXiv e-prints arXiv:2309.12300*, Sept. 2023.

[27] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations," *arXiv e-prints arXiv:2303.01497*, Mar. 2023.

[28] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick, "Dreamitate: Real-World Visuomotor Policy Learning via Video Generation," *arXiv e-prints arXiv:2406.16862*, June 2024.

[29] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," *arXiv preprint arXiv:2310.06114*, 2023.

[30] K. Pertsch, R. Desai, V. Kumar, F. Meier, J. J. Lim, D. Batra, and A. Rai, "Cross-Domain Transfer via Semantic Skill Imitation," *arXiv e-prints arXiv:2212.07407*, Dec. 2022.

[31] K. Grauman, A. Westbury, L. Torresani, K. Kitani, J. Malik, T. Afouras, K. Ashutosh, V. Baiyya, *et al.*, "Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives," *arXiv e-prints arXiv:2311.18259*, Nov. 2023.

[32] J. Urain, A. Mandlekar, Y. Du, M. Shafiullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters, "Deep Generative Models in Robotics: A Survey on Learning from Multimodal Demonstrations," *arXiv e-prints*, p. arXiv:2408.04380, Aug. 2024.

[33] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak, "Affordances from Human Videos as a Versatile Representation for Robotics," *arXiv e-prints arXiv:2304.08488*, Apr. 2023.

[34] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel RealSense Stereoscopic Depth Cameras," *arXiv e-prints arXiv:1705.05548*, May 2017.

[35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Academic Press, 2002.

[36] T. Yenamandra, F. Bernard, J. Wang, F. Mueller, and C. Theobalt, "Convex Optimisation for Inverse Kinematics," *arXiv e-prints arXiv:1910.11016*, Oct. 2019.

[37] L. Medeiros *et al.*, "Lang-segment-anything," https://github.

com/luca-medeiros/lang-segment-anything, 2023, accessed: 2024-09-15.

[38] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, “Segment Anything,” *arXiv e-prints arXiv:2304.02643*, Apr. 2023.

[39] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging Properties in Self-Supervised Vision Transformers,” *arXiv e-prints arXiv:2104.14294*, Apr. 2021.

[40] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, “Cotracker: It is better to track together,” *arXiv preprint arXiv:2307.07635*, 2023.

[41] G. E. Uhlenbeck and L. S. Ornstein, “On the theory of the brownian motion,” *Phys. Rev.*, vol. 36, pp. 823–841, Sep 1930. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.36.823

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint*, 2015.

[43] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, “Mastering visual continuous control: Improved data-augmented reinforcement learning,” *arXiv preprint arXiv:2107.09645*, 2021.

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.