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

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

New York University [*]

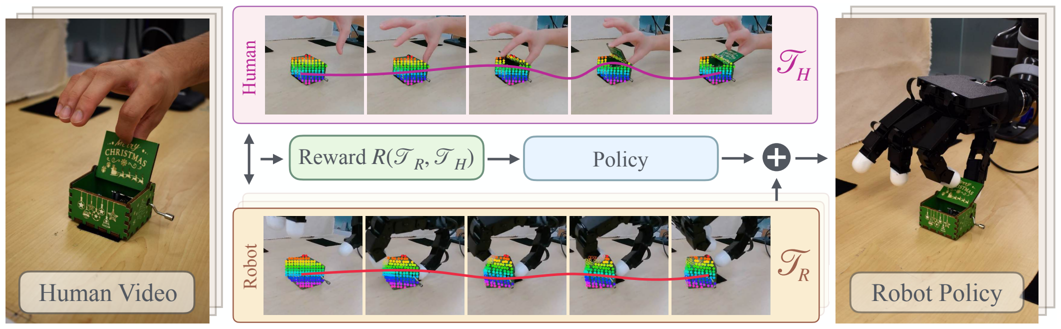

Figure 1: HUDOR generates rewards from human videos by tracking points on the manipulable object, indicated by the rainbow-colored dots, over the trajectory. This allows for online training of multi-fingered robot hands given only a *single* video of a human solving the task (left) and without robot 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: We introduce HUDOR, a technique that facilitates learning autonomous tasks for multi-fingered robot hands from human videos. HUDOR enables online fine-tuning by constructing a reward function from human videos using object-oriented signals from point trackers. This allows meaningful learning even with the robot hand in view. HUDOR enables a four-fingered Allegro hand to learn tasks from a single human video in just one hour of online interaction, outperforming alternatives by 4× on average across four tasks. Code and videos are available on our website `https://object-rewards.github.io/`.

Keywords: Human-to-Robot, Dexterous Manipulation

1 Introduction

Humans perform a wide range of dexterous tasks effortlessly in daily life [1]. Achieving similar capabilities in robots is crucial for their real-world deployment. Recent advances have enabled multimodal, long-horizon dexterous behaviors for two-fingered grippers [2, 3, 4, 5] through imitation learning (IL) from teleoperated robot data. However, extending these methods to multi-fingered hands is challenging because of high data requirements for robustness, especially in precision-demanding tasks, and the difficulty of teleoperating multi-fingered hands due to the need for low-latency, continuous feedback [6, 7, 8]. An alternative is learning from first-person videos of humans [9, 10, 11]. However, most approaches still rely on teleoperated demonstrations [12] or human intervented learning [13] to bridge the gap between human and robot hands.

[*][†]Correspondence to `irmakguzey@nyu.edu`.

We present HUDOR, an online imitation learning approach that bridges human videos and robot policies. An initial robot replay, generated using pose transformation and inverse kinematics from human demonstrations, often fails due to differences in human and robot hand morphology. HUDOR improves this by: (a) tracking target object points in human and robot videos, (b) calculating object motion and articulation similarity, and (c) fine-tuning the robot's trajectory via inverse reinforcement learning. Through iterative refinement, the robot adapts human actions to its own physical constraints.

We evaluate HUDOR on four challenging dexterous tasks. Our contributions includes:

1. HUDOR introduces the first framework that enables the learning of dexterous policies on multi-fingered robot hands using a single human video and hand pose trajectory (Section 3).
2. HUDOR introduces a new object-oriented reward calculation method that matches human and robot trajectories.
3. HUDOR outperforms state-of-the-art offline imitation learning methods for learning from human demonstrations [13, 3], achieving an average improvement of 2.64× (Section 3.1).

2 Learning Teleoperation-Free Online Dexterious Policies

HUDOR 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.

2.1 Robot Setup and Human Data Collection

Our hardware setup includes a Kinova JACO 6-DOF robotic arm with a four-fingered Allegro hand [6] attached, two RealSense RGBD cameras [14] and a Meta Quest 3 VR headset used to collect hand pose estimates. We compute 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 to compute relative transformations between camera frames. Details of relative transformations used and data alignment can be found in appendix (Section 5.2).

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. 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. [15]. The hand learning rate is set to be 50 times higher than the arm learning rate, allowing the IK to prioritize the hand movements.

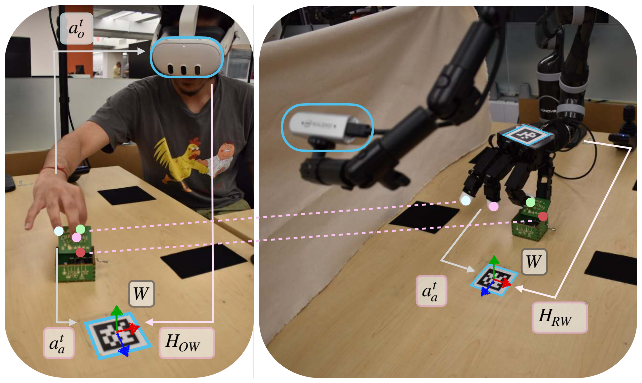

Figure 2: Robot setup and trajectory transfer in HUDOR. The VR headset is used solely for obtaining hand pose estimates and can be worn or attached to the setup as needed. World frame W is visualized on the ArUco marker on the operation table.

2.2 Residual Policy Learning

Due to morphological differences between human and robot hands, and VR hand pose estimation errors, naively replaying retargeted fingertip trajectories on the robot often fails to solve tasks. To address this, we employ an online residual policy using inverse reinforcement learning (IRL) to enhance trajectory replay. Given the visual disparity between human and robot hands, reward functions based on image-based matching [16, 17] with in-domain demonstration data are ineffective in providing reward signals. To bridge this domain gap, we introduce a novel object-centric trajectory-matching reward algorithm.

Object Point Tracking and Trajectory Matching We use off-the-shelf computer vision models to track the motion of points on the object of interest for reward computation. 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 appendix (Section 5.3), we explain our reward calculation in detail.

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 [18, 19] to ensure smooth robot actions. We learn a residual policy $\pi_r(\cdot)$ on this subset by maximizing the reward function R_i^{H2R} for each episode i using DrQv2 [20].

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: $a^t = a_r^t + \pi_r(a_r^t, \Delta s^t, \mathbb{E}(P_R^t), \mathcal{T}_R^t)$. The action, a^t, is sent to the IK module which converts it into joint commands for the robot.

3 Results

We experiment with four dexterous tasks, namely bread picking, card sliding, music box opening, and paper sliding, which are visualized in Figure 4. Exploration axes mentioned are with respect to the base of the robot.

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.

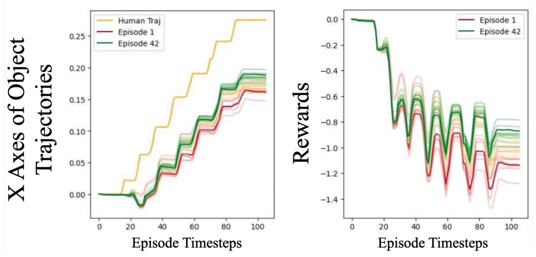

Figure 3: Illustration of how online correction improves robot policy and moves the robot trajectory closer to the expert's as time progresses in the Paper Sliding task. We showcase the X-axis of the trajectory of the tracked points for different episodes and their corresponding rewards.

3.1 How important are online corrections?

Figure 3 demonstrates how online learning improves the policy in the Paper Sliding task. 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 other offline baselines, with details can be found in appendix (Section 5.4).

Table 1 shows the comparison results. 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 model fails due to out-of-distribution data. More of the failure cases can be seen on our website. 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.

3.2 Does HUDOR improve over other reward functions?

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 Image OT and Pred OT. For Image OT [16], We pass RGB images from both the robot and the human videos through pretrained Resnet-18 [21] image encoders to get image representations and apply optimal transport (OT) based matching on them to get the reward, similar to FISH [16]. For 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 τ_r^p and τ_h^p.

Table 1: 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 [13]	3	0	0	12 ± 1.3
HUDOR (ours)	**8**	**7**	**6**	$\mathbf{17.3 \pm 1.5}$

Table 2: Comparison of success of HUDOR to different reward extraction algorithms.

Method	HUDOR(ours)	Image OT [16]	Pred OT
Bread Picking	**8**	6	6
Music Box Opening	**6**	1	2
Paper Sliding (cm)	$\mathbf{17.25 \pm 1.47}$	16.1 ± 1.37	16.5 ± 1.23

We present the success rates in Table 2. Further discussion and evaluation of point-tracking-based rewards can be found in the appendix (Section 5.5).

3.3 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 appendix. 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. These experiments show how point-tracking can enable some policy generalization.

4 Conclusion

This paper introduces HUDOR, a point-tracking, object-oriented reward mechanism aimed at improving human-to-robot policy transfer for dexterous hands. HUDOR outperforms both offline and online methods with varying reward functions and generalizes to new objects.

However, HUDOR has three limitations: it relies solely on in-scene human videos, limiting its generalization; it requires prior knowledge of suitable action dimensions for exploration; and it lacks a retry mechanism within episodes, making long-term training difficult. Enhancing HUDOR with in-the-wild data collection and a multi-stage learning framework could address these challenges.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

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)*, pages 378–383, 2016. doi:10.1109/ICRA.2016.7487156.

[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. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[6] S. Pandian Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-Dex: Teaching Dexterity with Immersive Mixed Reality. *arXiv e-prints arXiv:2210.06463*, Oct. 2022.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36: 823–841, Sep 1930. doi:10.1103/PhysRev.36.823. URL `https://link.aps.org/doi/10.1103/PhysRev.36.823`.

[19] T. Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

[21] 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*, pages 770–778, 2016.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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*, art. arXiv:2408.04380, Aug. 2024. doi:10.48550/arXiv.2408.04380.

[34] 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.

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

[36] L. Medeiros et al. Lang-segment-anything. `https://github.com/luca-medeiros/lang-segment-anything`, 2023. Accessed: 2024-09-15.

[37] 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.

[38] 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.

[39] 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.

[40] 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*, pages 652–660, 2017.

5 Appendix

5.1 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.

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 [22, 23, 6]. Some works have addressed this problem by training policies in simulation and deploying them in the real world [24, 23]. Although this approach has produced impressive results for in-hand manipulation [25, 26], closing the sim-to-real gap becomes cumbersome when manipulating in-scene objects.

Other works have focused on developing different teleoperation frameworks [27, 28, 7**?**] 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 [**?** 17], either by using nearest-neighbor matching for action retrieval [**?** 7, 27] or by initializing with a single demonstration and learning a residual policy with online interactions to improve generalization [17, 16].

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 [29, 30, 31, 32]. Some efforts focus on learning simulators that closely mimic the real-world environment of the robot from human videos using generative models [30, 33, 29], 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 [31, 34]. However, these works either require low-level policies to learn action primitives for interacting with objects [31], or only focus on simple tasks where a single point of contact is sufficient for manipulation [34]. Yet other approaches leverage on-scene human videos to learn multi-stage planning [12, 10], 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 [13], 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.

5.2 Human Data Collection

Relative Transformations We collect human hand pose estimates using existing hand pose detectors on Quest 3, and capture visual data using the RGBD cameras. We use two ArUco markers 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.

Fingertip pose for the i^{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 roboxt 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:

$$a_r^{t,i} = H_{RW}^{-1} \times a_w^{t,i} \tag{1}$$
$$= H_{RW}^{-1} \times H_{OW} \times a_o^{t,i} \tag{2}$$

where, $a_r^{t,i}$ are the homogeneous coordinates of the i^{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.

Data aglinment 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.

5.3 Object Point Tracking and Trajectory Matching

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 [36, 37] – `langSAM`. `langSAM` uses a text prompt and GroundingDINO [38] to extract bounding boxes for the object, which are then input to the SAM [37] to generate a mask. The output of `langSAM` corresponding to o^1 is a segmentation mask 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 [39]. Given a trajectory of RGB images, τ, 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 τ for all $t \in \{1 \ldots T\}$, where $P^1 = [p_1^1, \ldots p_N^1]$. We use $\tau^p = [P^1, \ldots P^T]$ 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, δ_{trans}^t, and mean rotation at time t, δ_{rot}^t. δ_{trans}^t is defined as the mean displacement of all the points in P^t from P^1. Similarly, δ_{rot}^t is defined as the mean rotation vector about the centroid of all the points in P^t from P^1. Concretely,

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

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

We define the *object motion* at time t as $\mathcal{T}^t = [\delta_{trans}^t, \delta_{rot}^t]$. Given two separate point trajectories, one corresponding to the robot τ_R^p and one corresponding to the human τ_H^p, 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:

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

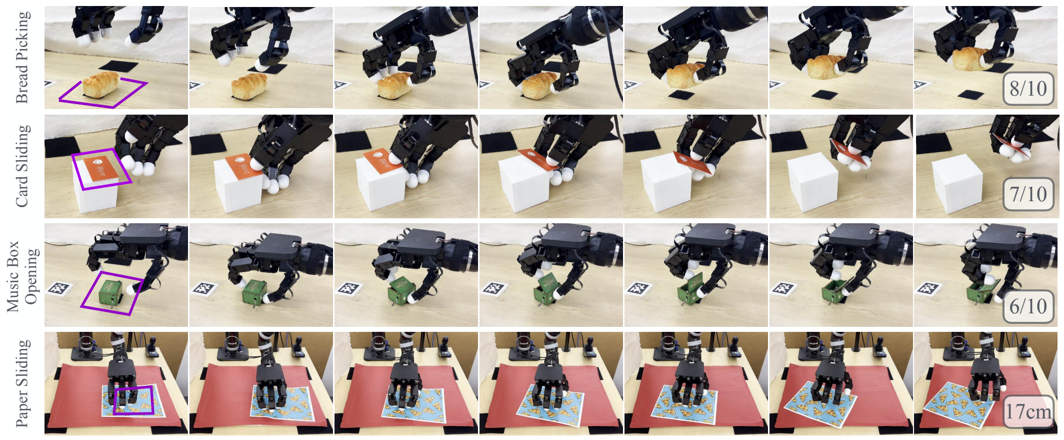

Figure 4: Rollouts of trained policies from HUDOR on four tasks are shown. For all tasks, training is conducted using a single human video. Success for each task is shown in the rightmost frames. Videos are best viewed on our website: `https://object-rewards.github.io/`.

5.4 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.
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 [13], 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 [40] 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.

5.5 Advantages of Trajectory Matching Reward

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.

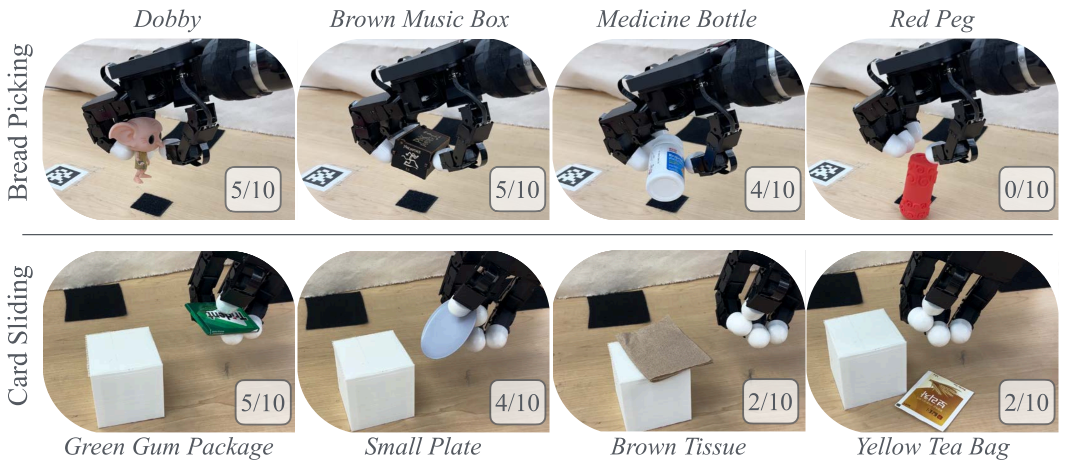

Figure 5: Generalization experiments on Bread Picking and Card Sliding task. We input the text prompts on top and bottom as input to the language grounded SAM model to get the initial mask for each object.