GenH2R: Learning Generalizable Human-to-Robot Handover via Scalable Simulation, Demonstration, and Imitation Supplementary Material

The supplementary material offers additional details on various aspects of the method and experiments. Refer to the table of contents below for an overview. Section A provides additional details and clarifications on our methods. Sections B and C present supplementary experiments on baselines, along with additional quantitative and qualitative results in the simulation and real-world scenarios, respectively. Section D discusses the limitations of our work and explores potential research directions for future human-to-robot handovers and human-robot interactions.

Contents

A. More Method Details

A.1. GenH2R-Sim

In this section, we provide details on the generation of hand-object moving trajectories and our simulator GenH2R-Sim.

A.1.1 Hand-Object Moving Trajectory Generation

Note: The unit for all positions in the following paragraphs is meters.

In HandoverSim [9] and GenH2R-Sim, the robot arm's base is located at $(0.61, -0.50, 0.875)$, and the center of the table surface is at $(0.61, 0.28, H = 0.92)$. To synthesize a hand-object moving trajectory, we start by generating the object's trajectory.

For the translation part, we uniformly sample a starting point from the starting region $[0.3, 0.9] \times [0, 0.2] \times [H + 0.1, H + 0.3]$. Subsequently, we sample several endpoints from the activity region $[0.1, 1.1] \times [-0.3, 0.1] \times [H + 0.1, H + 0.7]$ and employ Bézier curves to connect the starting point and the endpoints. For each Bézier curve, a key point is sampled from a Gaussian distribution centered at the midpoint with a standard variation of 0.2, and the translation speed along this curve is uniformly sampled from $[0.1, 0.2]\ \mathrm{m\,s^{-1}}$.

For the rotation part, we uniformly sample a rotation $R \in \mathrm{SO}(3)$ as the starting object orientation. When the object travels along a Bézier curve, we rotate the object about a random rotation axis with an angular speed uniformly sampled from $[0.5, 1]\ \mathrm{rad\,s^{-1}}$.

After generating the object trajectory $\xi = (\mathcal{T}_0, \mathcal{T}_1, \cdots, \mathcal{T}_{T-1})$, we correspondingly generate the hand pose trajectory $\varsigma = \xi \circ T^{\text{hand}}_{\text{object}}$, where $T^{\text{hand}}_{\text{object}}$ is the hand pose in the object reference frame.

A.1.2 More Details about GenH2R-Sim

In real-world handovers, in order to enhance stability, our motion typically stops when another person's hand is close to the object. We incorporate this characteristic into our simulator, making it reactive to the robot arm's motion. To be specific, suppose $p \in \mathbb{R}^3$ is the current position of the gripper's tip, and $Q \subset \mathbb{R}^3$ is the current object point cloud. If $\min_{q \in Q} \|p - q\| \leq 0.1$, the hand and object will stop moving, awaiting the robot arm to grasp the object. We believe that this modification makes the cooperative handover process more realistic, transforming it from a simple chase-and-grasp game into a more authentic interaction.

It is important to note that we apply this modification exclusively in our simulation environment, GenH2R-Sim, for the benchmarks "t0" and "t1". We refrain from modifying it in HandoverSim [9] for the benchmark "s0" to ensure a fair comparison with the exact results obtained by baseline methods.

A.2. Generating Demonstrations for Distillation

A.2.1 Clarification of Different Methods

We would like to provide further clarification regarding the terms used to describe various demonstration generation methods, as outlined in Table 3 of the manuscript. These terms include **destination planning**, **dense planning**, and **landmark planning**.

Destination planning refers to the foresighted planner discussed in Section 3.3, which plans directly to the object's destination at the beginning. While the generated demonstrations exhibit smoothness, they lack vision-action correlation in complex scenarios and are not distillation-friendly.

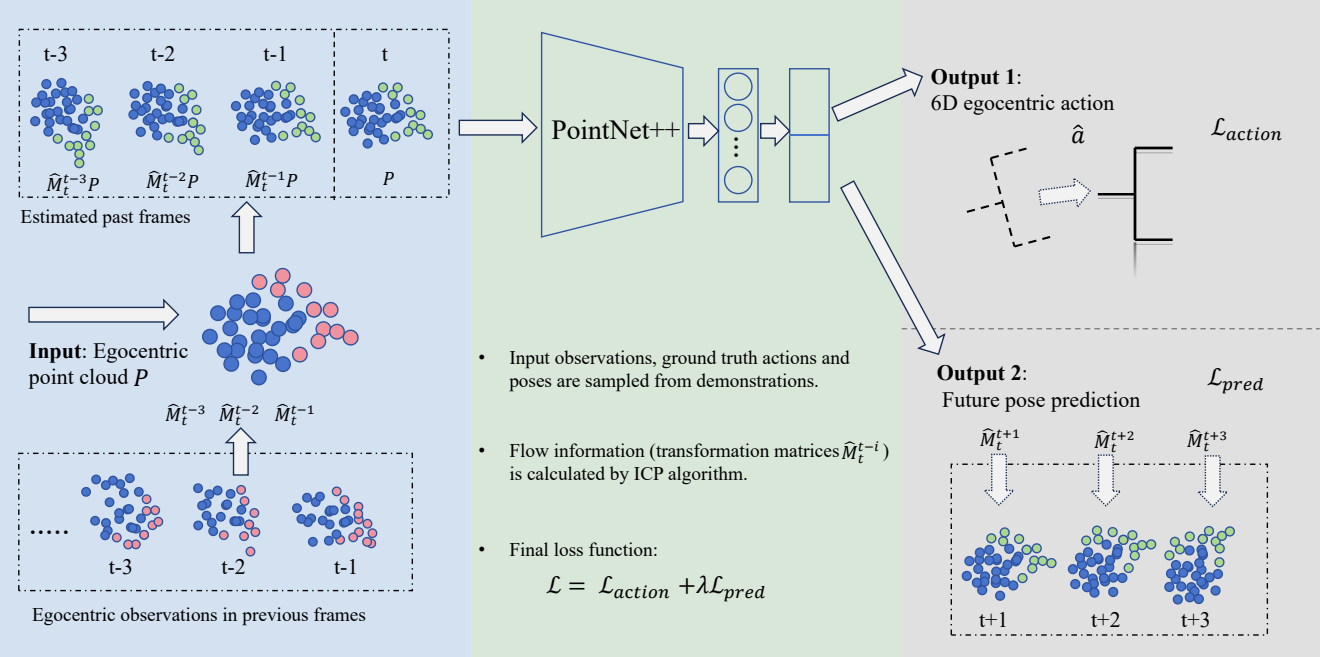

Figure 5. **Forecast-Aided 4D Imitation Learning Pipeline.** The network receives egocentric point cloud input and produces egocentric 6D actions as output. For each input point, we compute its past coordinates using flow information obtained through the Iterative Closest Point algorithm. Subsequently, we employ PointNet++ to encode the processed point cloud into a low-dimensional global feature. The policy head decodes this feature into a 6D egocentric action, serving as the primary policy output. Simultaneously, the prediction head decodes the feature into future pose transformations, contributing to the auxiliary loss.

Dense planning represents the special case of our method, where the planner plans at each time step based on the current object position. Although the generated demonstrations ensure a strong vision-action correlation, they suffer from the zigzag issue, resulting in slower performance.

Landmark planning denotes our method, where the planner periodically plans based on the object's position at future landmarks.

A.2.2 Trajectory Resampling Strategy

In this section, we introduce another resampling strategy designed to enhance the vision-action correlation in expert demonstrations.

Upon successfully solving the Inverse Kinematics (IK) to achieve the 6D target grasping pose, the subsequent task involves generating a trajectory from the initial 7D joint configuration C to the destination 7D joint configuration D. Through optimization, OMG-Planner [42] produces a trajectory $C_0 = C, C_1, \cdots, C_{N-1}, C_N = D$ with fixed time steps N. However, a challenge arises as the step size of the expert action is influenced by the distance between the initial end effector pose and the target grasping pose. Specifically, if the target grasping pose is far away from the initial end effector pose, the expert will move faster; conversely, if the target grasping pose is close to the initial end effector pose, the expert will move slower. This variability in step size can potentially confuse the vision-based model, which lacks awareness of the initial end effector pose and struggles to discern the expert's speed.

To address this issue, we conduct a resampling process to refine the obtained trajectory. Let $s_i = \sum_{j=1}^{i} \|C_j - C_{j-1}\|$ represent the accumulated step length, and L denote the hyperparameter controlling the desired step length. The resampled trajectory, denoted as $C'_0 = C_0, C'_1, \cdots, C'_{M-1}, C'_M = C_N$, where $(M-1)L < s_N \leq ML$, and for each $1 \leq i \leq M-1$, suppose $s_j \leq iL \leq s_{j+1}$, then

$$C'_i = \frac{s_{j+1} - iL}{s_{j+1} - s_j} C_j + \frac{iL - s_j}{s_{j+1} - s_j} C_{j+1}. \tag{3}$$

The resampling ensures that, regardless of the proximity between the initial end effector pose and the target grasping pose, the resulting trajectory maintains a consistent step length. This characteristic makes the expert demonstrations more conducive to distillation for vision-based models.

A.3. Forecast-Aided 4D Imitation Learning

Figure 5 provides an overview of our Forecast-Aided 4D Imitation Learning process. Similar to Handover-Sim2real [11], We initiate the pipeline by obtaining an ego-

		s0 (Sequential)			s0 (Simultaneous)			t0			t1		
		S	T	AS	S	T	AS	S	T	AS	S	T	AS
	OMG Planner† [42]	62.50	8.31	22.5	-	-	-	-	-	-	-	-	-
train on s0	GA-DDPG [43]	50.00	**7.14**	22.5	36.81	**4.66**	23.6	23.59	7.31	10.3	46.7	**5.50**	26.9
	Handover-Sim2real [11]	75.23	7.74	**30.4**	68.75	6.23	35.8	29.17	6.29	15.0	52.40	7.09	23.8
	Handover-Sim2real* [11]	64.35	7.61	26.7	25.69	5.43	15.0	28.56	4.73	17.9	30.60	5.98	16.5
	Destination Planning	74.31	9.01	22.8	76.16	6.98	35.2	25.68	5.96	14.1	48.4	8.94	15.1
	Dense Planning	74.77	9.54	19.8	75.45	7.32	33.0	27.30	6.26	14.1	52.3	9.24	15.1
	Landmark Planning	77.78	9.24	22.3	79.17	7.26	34.9	29.63	6.23	15.4	54.2	9.02	16.6
train on t0	GA-DDPG [43]	54.76	7.26	24.2	44.68	5.30	26.5	24.05	4.70	15.3	25.50	5.86	14.1
	Handover-Sim2real [11]	65.97	7.18	29.5	62.50	6.04	33.5	33.71	5.91	18.4	47.10	6.35	24.1
	Handover-Sim2real* [11]	63.55	7.58	26.5	38.89	5.29	23.1	33.31	**4.64**	21.4	33.35	5.81	18.4
	Destination Planning	0.93	12.80	0.01	6.48	12.41	0.3	5.96	8.81	1.9	1.60	12.03	0.1
	Dense Planning	81.48	9.51	21.9	84.95	7.45	36.3	38.04	7.16	17.1	57.90	8.85	18.4
	Landmark Planning	**86.57**	8.81	28.0	**85.65**	6.58	**42.8**	**41.43**	6.01	**22.3**	**68.33**	7.70	**27.9**

Table 4. **Evaluating on different benchmarks.** We compare our method against baselines from the test set of HandoverSim [9] benchmark ("s0 (sequential)" and "s0 (simultaneous)") and our GenH2R-Sim benchmark ("t0" and "t1"). We use the best-pretrained models from the repositories of GA-DDPG [43] and Handover-Sim2real [11] for evaluation. The results for our method are averaged across 3 random seeds. Note that S means success rate(%). T means time(s). AS means average success(%) as defined in Equation 2. †: This method [42] is evaluated with ground-truth states and cannot handle dynamic handover like "s0 (Simultaneous)", "t0" and "t1".*: We reproduce the results of HandoverSim2real in the true simultaneous setting as detailed in Section 4.1 to make a fair comparison.

centric hand and object point cloud from the simulator, together with the one-hot encoding for hand/object labels. We augment the feature of each point with its 3D coordinates in the past n time steps, computed from the estimated flow information and the current 3D coordinates. As a result, each point has a feature vector of length $3 + 2 + 3 \cdot n$.

We then introduce the specific method for computing flow information. Given the end effector pose, we convert the point cloud from the egocentric frame to the static world frame and store it in a buffer. In each time step, we retrieve the point clouds of several past time steps and leverage the Iterative Closest Point (ICP) registration algorithm [37] to estimate transformation matrices between the current point cloud and past point clouds in the world frame. While these transformations may be slightly imprecise due to the incomplete point cloud input, they can provide sufficient flow information for each point. Finally, the flow information is converted back to the current egocentric frame, which serves as an important part of our feature representation.

Then we feed the point cloud with processed features into PointNet++ [35] to obtain a global low-dimensional representation. This representation is then decoded by two heads: the policy head and the prediction head. The policy head decodes it into a 6D egocentric action, and $\mathcal{L}_{action}$ is computed following the approach defined in [27]. Simultaneously, the prediction head decodes the representation into transformations between the current and future object poses, with $\mathcal{L}_{pred}$ computed as a motion prediction loss.

Similar to Handover-Sim2real [11], our policy only outputs 6D egocentric actions in a closed loop. For decisions on whether to grasp the object and place it in the target location, we adopt a heuristic method akin to GA-DDPG [43]. Specifically, if the number of points in the gripper's vicinity exceeds a predefined threshold, the robot attempts to grasp the object and retract to the target location in an open-loop fashion, foregoing the execution of the egocentric actions predicted by the policy network.

B. Simulation Experiments Details

B.1. Discussions on the Simultaneous Setting

It's essential to clarify that in handoverSim [9] and Handover-Sim2real [11], the simultaneous setting (also referred to as "w/o hold") implies that the robot is allowed to move from the beginning of the episode. **We adhere to this definition in our GenH2R-Sim and our methods**. In the settings "s0 (Simultaneous)", "t0", and "t1", the robot initiates movement immediately upon detecting the object.

However, we observed that in the code of Handover-Sim2real, a parameter named "TIME_WAIT" is used to specify the time to wait before executing the actual action in different settings. In "s0 (Sequential)", "TIME_WAIT" is set to 3s (matching the 3-second duration of DexYCB [8] handover motion), but in "s0 (Simultaneous)", "TIME_WAIT" is set to 1.5s, which implies a 1.5s wait for the simultaneous setting. We believe this value should be 0s for the simultaneous setting.

The author adjusted this parameter after observing that humans typically move faster than the robot. This modification aims to prevent collisions, especially when the robot approaches the human while the human is also approaching the object. The change is implemented to reduce the num-

ber of failures caused by attempting to grasp while the human is still in motion or before the human completes picking up the object from the table.

We believe that improving performance makes sense, but we consider the true simultaneous setting to be closer to real-world scenarios. It's crucial not to make the person wait for an extended period during a short-term handover process, so we avoid adjusting the waiting time manually.

For a fair comparison, we reproduce the results in the true simultaneous setting.

As highlighted in the blue rows of Table 4, our method demonstrates significant improvements. When trained on "s0", our method achieves improvements of 13.43%, 53.48%, 1.07%, and 23.6% in the successful rate of "s0 (Sequential)", "s0 (Simultaneous)", "t0", and "t1" settings, respectively. When trained on "t0", our method achieves improvements of 23.02%, 46.76%, 8.12%, and 34.98% across the four test sets. Notably, our method excels in the simultaneous setting when hands are in motion. This highlights that our distillation-friendly demonstrations can better extract valuable insights from more complex scenarios and showcase enhanced generalizability compared to the baseline.

B.2. Baseline Experiments

When training Handover-Sim2real, we follow the two-stage teacher-student training approach outlined in the paper. In the pretraining stage, the duration of hand and object movement is clipped to 3 seconds, and the expert waits for 3 seconds before planning to grasp the static object. In the finetuning stage, the entire movement is used. In the original codebase, the robot waits 1.5 seconds. In our reproduced version, the robot does not need to wait, moving directly with the dynamic hand and object.

B.3. More Ablations

We conducted additional ablations for our method, as shown in Table 5. All these methods are trained in "t0" and tested in "t0".

In Section 3.4 of the manuscript, we mentioned that frame stacking is a straightforward approach but struggles to capture both motion and geometry effectively. To quantitatively demonstrate this, we compared it with our method based on forecast-aided 4D imitation learning and found a 5.26% decrease in the success rate. This highlights the effectiveness of our method in learning from 4D spatial-temporal information.

Moreover, excluding the endpoints from consideration when selecting the landmark results in a 1.70% decrease in the success rate. This indicates the importance of incorporating the endpoints when choosing the landmark state.

Methods	S	T	AS
w/o Flow	31.66	**5.67**	17.9
w/o Prediction	39.18	6.11	20.7
w/o Flow & Prediction	37.04	5.93	20.1
w/o Endpoints	39.73	5.90	21.7
Frame Stacking	35.17	5.82	19.4
Ours	**41.43**	6.01	**22.3**

Table 5. **Ablations on different modules.** "w/o Flow" means not using flow information in the input. "w/o Prediction" means not adding prediction loss in the final loss.

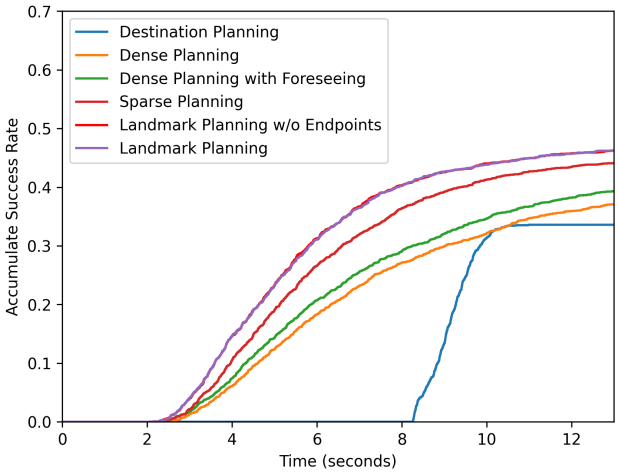

Figure 6. **Comaprison of different expert demonstrations.**

B.4. Evaluating on Demonstration Generation

Figure 6 compares different expert demonstration generation variants by showing their accumulated success rate w.r.t. to success time on "t0".

Destination Planning is suboptimal, as it plans a straight trajectory directly to the destination, which is not only slow but also nearly impossible to be distilled to vision-based agents when the object trajectory is complex.

With access to future object states and with less frequent replanning periods, Landmark Planning exhibits a larger success rate and faster success time compared with Dense Planning. To ablate the two factors, we also analyze two additional settings. In Dense Planning with Foreseeing, OMG-Planner still always replans at each step, but based on the future object states. In sparse Planning, OMG-Planner replans with the same sparsity as Landmark Planning, but it can only plan based on the current object state. We can analyze from the curves that both decreasing the replanning period and foreseeing future states can help the expert to achieve a higher success rate and lower success time.

Figure 7. **Real-world Handover System Setup.** Our system consists of an xMate 3 robot, which is similar to a Franka Panda robot, and two RealSense Depth Camera D435 devices.

B.5. Training Details

In our training process, we employ a single Nvidia GeForce RTX 3090 (24GB) with a batch size of 256. The training spans 80,000 iterations, with each iteration involving the random sampling of 256 observation-action pairs from demonstrations. We use the Adam optimizer with a learning rate of 0.001 and weight decay of 0.0001. The entire process takes approximately 8 hours to train our method. We incorporate flow information from the last 3 time steps and calculate the prediction loss for the next 3 time steps with the weighting hyper-parameter λ as 0.1.

C. Real World Experiments Details

C.1. Setup

In our real-world handover system, illustrated in Figure 7, we utilize a ROKAE xMate 3 ER series flexible collaborative robot along with two Intel RealSense Depth Camera D435 devices.

Since the action space is 6D Cartesian, while our policies and the baselines are trained with Franka Panda robot in HandoverSim [9] and GenH2R-Sim, we can deploy them to the xMate3 robot despite their morphology difference, as they have own position controllers. Thus, our real-world experiments not only assess the policy's generalization for sim-to-real transfer but also evaluate its adaptability to different robots.

Our real-world handover system incorporates two Intel RealSense Depth Camera D435 devices to enhance the egocentric point cloud. In Figure 7, the higher camera captures the point cloud in proximity to the robot gripper but lacks visibility further ahead. Conversely, the lower camera captures the point cloud ahead of the robotic arm but misses details near the gripper. By merging the point clouds from both cameras, we achieve a comprehensive view, which is beneficial for the deployed policies.

Figure 8. **Various objects for real-world handover.** The image above displays relatively simple objects for handover, such as the can, the box, the bottle, or some square objects. In contrast, the image below showcases more challenging objects for handover, including the plastic stool, the teapot, the sticky tape, or some soft objects with diverse shapes.

The robot initiates movement upon perceiving a point cloud. In case the object is not visible during the handover process, the robot tracks the object's last known pose. The baselines, GA-DDPG [43] and Handover-Sim2real [11], are treated similarly to [11]. For GA-DDPG, the pre-trained policy model is loaded, and heuristic methods determine whether to grasp. For Handover-Sim2real, both the pre-trained policy model and the pre-trained grasp prediction network are loaded.

C.2. User Study

The study involved 6 participants who compared our forecast-aided 4D imitation learning method based on landmark planning demonstrations, with two baseline methods, GA-DDPG and Handover-Sim2real. The first user study contains 5 different objects in two settings. The second user study contains 15 various and novel objects.

For the first user study, The selected 5 objects for eval-

	Simple setting			Complex setting		
	GA-DDPG[43]	Handover-Sim2real[11]	Ours	GA-DDPG[43]	Handover-Sim2real[11]	Ours
1. plastic mug	5 / 6	4 / 6	6 / 6	2 / 6	2 / 6	4 / 6
2. EFES bottle	5 / 6	4 / 6	6 / 6	4 / 6	3 / 6	5 / 6
3. Cheez-It box	1 / 6	4 / 6	4 / 6	3 / 6	1 / 6	4 / 6
4. sticky tape	3 / 6	3 / 6	5 / 6	3 / 6	3 / 6	4 / 6
5. Pringle can	4 / 6	2 / 6	6 / 6	1 / 6	1 / 6	4 / 6
total	18 / 30 (60%)	17 / 30 (57%)	**27 / 30 (90%)**	13 / 30 (43%)	10 / 30 (33%)	**21 / 30 (70%)**

Table 6. **User study for sim-to-real experiments.** Each method was evaluated by six individuals for every object in both the simple and complex settings. Failure scenarios included collisions with the human hand, dropping to the table, or exceeding the time limit ($T_{\text{max}} = 13$ seconds). Our method consistently outperformed the baselines in the real-world handover system in both simple and complex settings, aligning with the results observed in the simulation experiments.

Objects	GA-DDPG	Handover-Sim2real	Ours
1. Transparent bottle	1 / 6	2 / 6	**3 / 6**
2. Transparent beaker	1 / 6	0 / 6	**2 / 6**
3. Transparent cup	1 / 6	2 / 6	**4 / 6**
4. Regular cup	**5 / 6**	1 / 6	**5 / 6**
5. Glue stick	3 / 6	2 / 6	**6 / 6**
6. Large teacup	3 / 6	2 / 6	**4 / 6**
7. Blue ball	1 / 6	1 / 6	**4 / 6**
8. Small sponge	3 / 6	1 / 6	**6 / 6**
9. Tape measure	3 / 6	**4 / 6**	**4 / 6**
10. Large bucket	4 / 6	1 / 6	**5 / 6**
11. Stapler	**3 / 6**	2 / 6	**3 / 6**
12. Disinfectant	3 / 6	2 / 6	**4 / 6**
13. Packaging box	2 / 6	2 / 6	**4 / 6**
14. Saw	2 / 6	2 / 6	**5 / 6**
15. Book	4 / 6	2 / 6	**6 / 6**
Overall	43.3%	28.9%	**72.2%**

Table 7. **User study for sim-to-real-experiments in various and novel objects.** Each method was evaluated by six individuals for every object. Our method outperformed the baselines by a large margin.

uation include a mug, a bottle, a cracker box, a sticky tape, and a chips can. The cracker box is an object shown in DexYCB [8] trajectories, while the other 4 novel objects may exhibit more diverse geometries.

In the simple setting, users hand each object to the gripper in a straightforward manner. In the complex setting, users execute a relatively long and quick trajectory, involving both translations and rotations. For each specific object, we try to ensure that each participant executed a similar trajectory in the same setting for different methods, ensuring a fair comparison.

Table 6 provides a detailed breakdown of the results presented in Table 3 of the manuscript. Our method is compared with baselines across different settings, revealing a remarkable 34% improvement in the simple setting and a substantial 40% improvement in the complex setting from Handover-Sim2real. Notably, in the simple setting, our method demonstrates great generalizability to various objects, including new objects with different geometries or similar objects of different sizes. In the complex setting, our method exhibits smooth object tracking with predictive intention, resembling a more human-like approach to grasping handed objects. Further analysis will be elaborated in our accompanying video. It is noteworthy that Handover-Sim2real exhibits a lower success rate compared with GA-DDPG. One possible explanation is that the pre-trained grasp prediction network may not be as robust as heuristic methods in determining whether to grasp, which may not be able to generalize well to novel objects and potentially increase the sim-to-real gap.

For the second user study, we also expanded the real-world experiments comparing our method and baselines. 6 users participated by handing over **15 diverse and novel objects** varying in geometry, size, and transparency. Users provide various trajectories, occasionally adopting a less cooperative or adversarial manner. They blindly test each policy on identical trajectories. Our method consistently outperforms baselines as shown in Table 7. While adversarial behavior and weird object geometry lead to failures in baselines, our approach generalizes well and adeptly adjusts to the hand trajectory quickly. For failure cases, our method faces challenges with transparent objects due to corrupted depth. The final version will include detailed tests and elaborations.

C.3. Generalization Study

In addition to direct comparisons with baseline methods, we conducted numerous real-world handover experiments involving different trajectories and objects.

Figure 8 showcases two sets of objects used in our experiments. The simple set comprises regular objects similar to those used in DexYCB [8] or HandoverSim [9], which are easier to pass and grasp. The difficult set includes more challenging objects with diverse shapes and geometries. We introduced variations in human behavior, such as different grasping poses or handover trajectories.

As shown in Figure 9, our qualitative experiments reveal the robustness of our policy, crafted through extensive large-scale demonstrations and an imitation learning framework. Trained within the GenH2R-Sim environment, our policy showcases effective generalization across diverse objects and various handover scenarios.

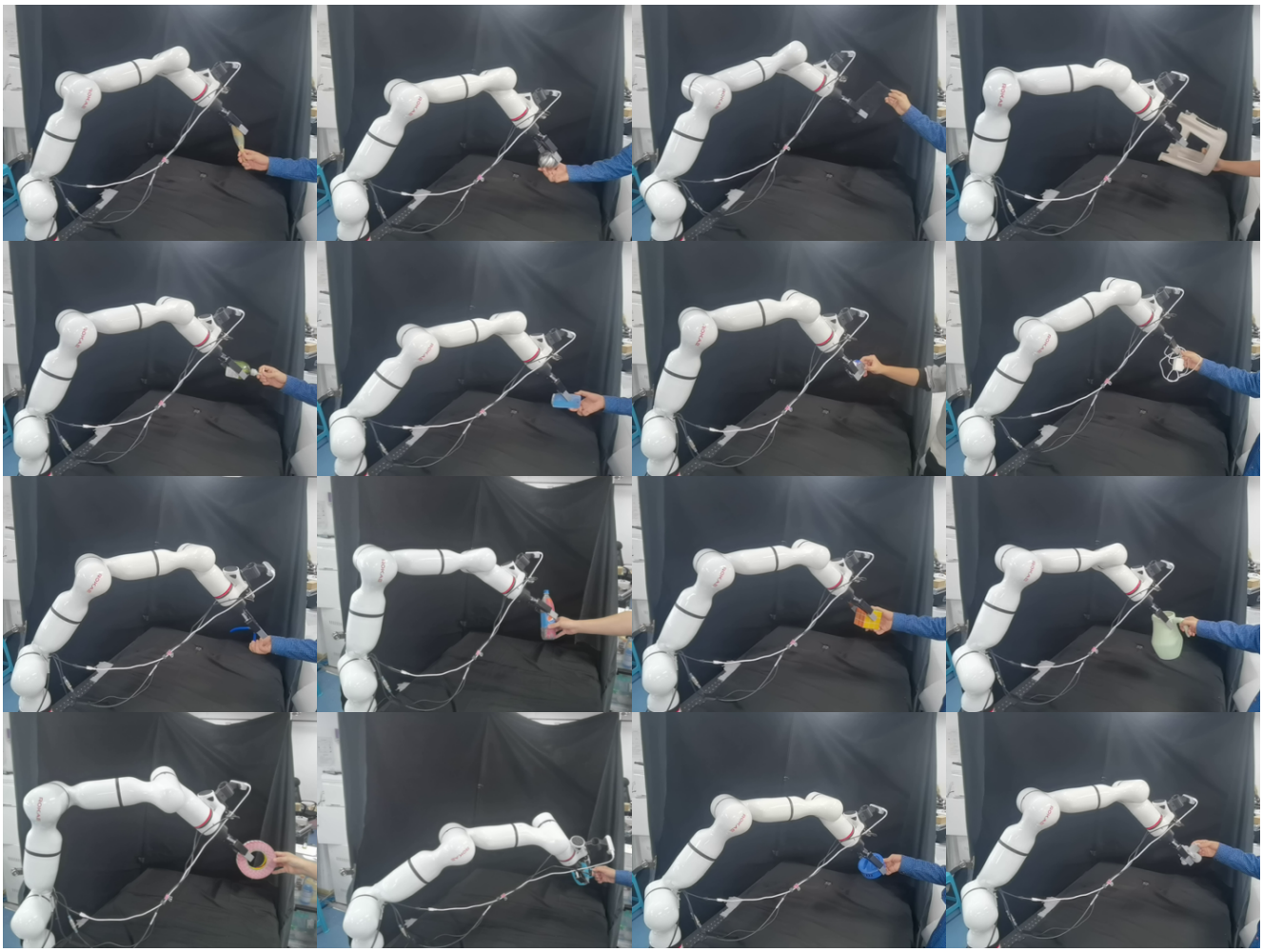

Figure 9. **Qualitative real-world results with various objects**. In the real-robot system, we qualitatively assess the generalization ability of our method by testing it with various objects.

D. Limitations and Future Work

While in this paper significant progress has been achieved in the H2R handover task, we acknowledge certain limitations that could serve as inspiration for exciting future research.

In aspects of robot morphology, we concentrate on the relatively simple 7DoF robotic arms, characterized by a confined activity region and limited motion capabilities. In contrast, robots equipped with a movable base exhibit a broader range of motion, enabling them to navigate and interact within a more extensive spatial environment, enhancing their versatility and efficacy in various human-robot interaction tasks.

In aspects of human modeling, our current focus on the object and hand poses neglects the consideration of the entire human body. In real-world scenarios, robots may need to take into account not only the hand pose and trajectories but also the motion of the entire human body for more dynamic and generalizable interactions. Extending the simulation environment to model a more complex representation of the human, including body movements, poses a challenging yet practical avenue for future work in policy learning.

In aspects of human intention, our simulator currently does not incorporate human intention. Existing simulation environments have mainly focused on physical modeling, lacking representation of human behavior. In HandoverSim [9], for instance, the human hand does not respond to the robot's actions. In GenH2R-Sim, we introduce a more interactive element, where the human hand stops moving and waits for handover when the robot arm is close to the object. However, there is room for more complex and interesting modeling of human behavior. For instance, when the gripper moves rapidly toward the hand, the human may perceive danger and retract the hand. Introducing more sophisticated representations of human behavior in the simulator is crucial for a human-centric handover process.