RoboWheel: A Data Engine from Real-World Human Demonstrations for Cross-Embodiment Robotic Learning

Supplementary Material

Contents

A. Notations

Symbol	Meaning	Notes / Space
$\{I_t\}_{t=1}^{T}$	Input frame sequence	RGB / RGB-D
K	Camera intrinsics	Known or estimated
T_c^w	Camera pose in world frame	$SE(3)$; often (R_{wc}, t_{wc})
$\Pi(\cdot)$	Pinhole projection operator	3D $\rightarrow$ pixel coordinates
$\mathbf{h}_t$	Hand state at time t	$(\theta_h(t),\ R_h^w(t),\ t_h^w(t))$
$\theta_h(t)$	Hand(MANO) parameters	Joint angles
$R_h^w(t),\ t_h^w(t)$	Wrist pose in world (rot/trans)	$SO(3)$ and $\mathbb{R}^3$
$\mathbf{p}_t$	Object state at time t	$T_o^w(t) \in SE(3)$
$T_o^w(t) = (R_o(t),\ t_o(t))$	Object pose in world (rot/trans)	$SE(3)$
$M_o,\ \hat{M}_o$	Object mesh (metric / up-to-scale)	Triangle mesh + texture
s_o	Object scale factor	From depth–mesh alignment
$m_t,\ D_t$	Object mask and depth map at t	Segmentation and depth
$P_t,\ P$	Back-projected points (per-frame / global)	From D_t and K
$\mathrm{AABB}(\cdot)$	Axis-aligned bounding box	Use diagonal via $\mathrm{diag}(\cdot)$
$\mathbf{A}$	Canonical action space	Unified by C2W and facing origin
$T_w^A = (R^{w\to A},\ t^{w\to A})$	World-to-action-space transform	Coordinate re-alignment
$\Delta^2(\cdot)$	Second-order temporal difference	Jitter suppression
$\rho(\cdot)$	Robust loss (e.g., Geman–McClure)	For reprojection errors
$d(\cdot,\cdot)$	Point–set / point–surface distance	For proximity/attraction
$V_h,\ V_o$	Hand mesh vertices / object surface points	Geometry sets
$\phi_h(\mathbf{x}; t)$	Hand signed distance field (SDF)	Positive outside (as used here)
$TSDF_o$	object truncated signed distance function	negative inside and zero else
$\tilde{\mathbf{q}}_i(t)$	World coords of sampled object surface point	$R_o(t)\,\mathbf{q}_i^{\mathrm{loc}} + t_o(t)$
$\mathcal{N}_t$	Near-contact candidate set	$\lvert\phi_h\rvert < \tau_{\mathrm{band}}$ or Top-K
$\lVert\cdot\rVert_F$	Frobenius norm	For rotation log etc.
$\log_{SO(3)}(\cdot)$	Lie-group log map on $SO(3)$	Rotation discrepancy
$d_{SE(3)}(\cdot,\cdot)$	Geodesic distance on $SE(3)$	Pose discrepancy
$\mathrm{E}_\pi[\cdot]$	Expectation under policy π	RL objective
r_t	Instantaneous reward	Weighted components
$\psi_{\mathrm{limits}}(\cdot)$	Joint-limit margin reward	Prefer mid-range
π_θ	Residual policy network	Added to a^{IK}
$T_{\mathrm{rel}}(t)$	Relative pose $T_h^{-1}(t)\,T_o(t)$	Hand–object relative pose
$R_g,\ p_g$	Gripper pose (rot/trans)	From hand keypoints
$q_t^{(r)}$	Joint configuration of robot r at t	From bounded-rate IK
$\phi_{\mathrm{lim}}(q)$	Joint-limit penalty	IK constraint term
$S = \mathrm{diag}(-1,1,1)$	Left–right mirror matrix	About sagittal plane
$R_y(\pi)$	Rotation about y by π	Used with S to set facing
$\varphi_{\mathrm{sem}}(\cdot)$	Semantic embedding (text–shape)	For semantic similarity
γ^t	Discount factor power	RL return discounting

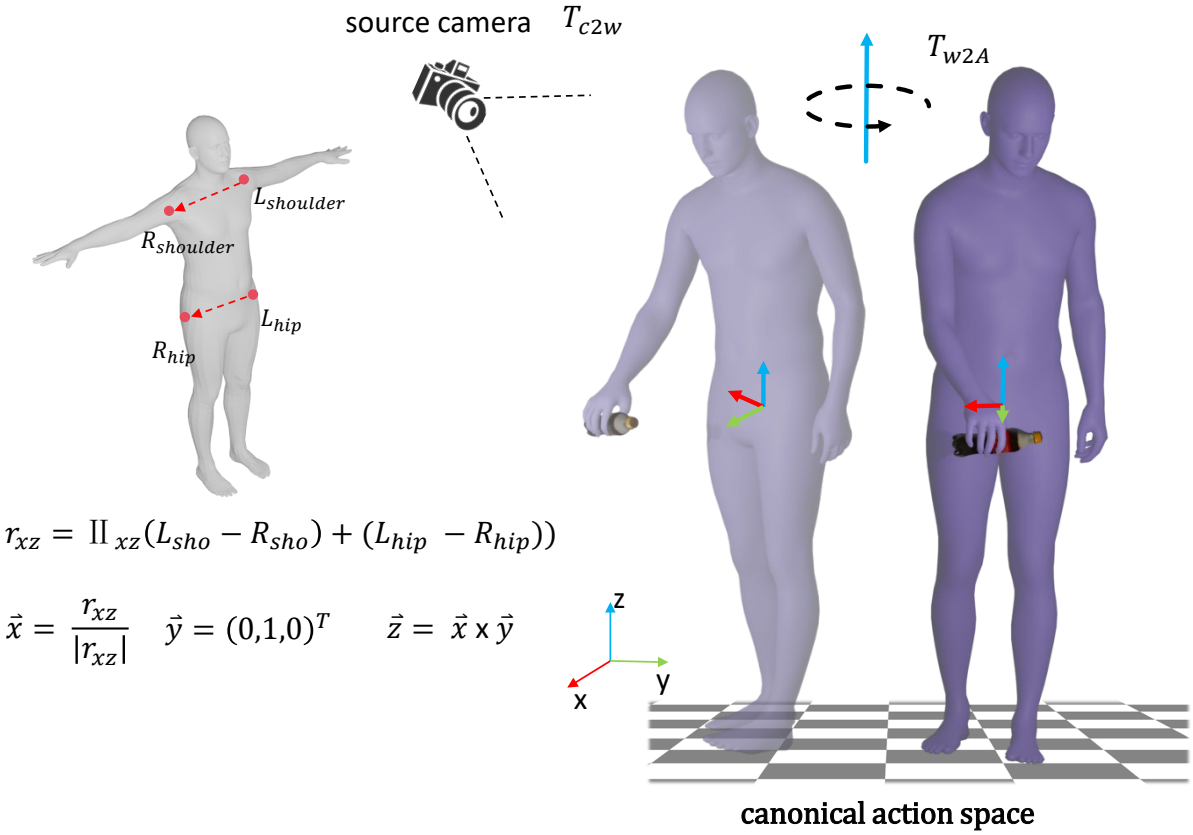

Figure 9. Our pipeline decouples the reconstructed hand-object motions from the specific camera viewpoint of the source video. We first lift the reconstructions to a consistent world frame using the camera-to-world transformation T_{c2w}. Subsequently, we normalize these trajectories into a canonical action space via $T_{w2\mathbf{A}}$. This two-step alignment ensures that the retargeted actions maintain a consistent approach direction and kinematic interpretation across all robotic embodiments.

B. Transform to Canonical Action Space

Real-world HOI videos are captured under arbitrary viewpoints, which leads to view-dependent reconstructions of both hand and object. To eliminate this inconsistency, we adopt a lifting procedure, as shown in Fig. 9. In the first step, all reconstructed HOI results are transformed from their respective camera coordinate systems into the same world coordinate system. In the second step, the trajectories in the world coordinate system are further normalized into a unified canonical action space, ensuring that interaction trajectories from heterogeneous sources become retargetable.

Step 1: Estimate (K, T_c^w) and lift to the world frame. We assume a static-camera prior and estimate camera intrinsics K and a (time-invariant) camera-to-world transform $T_c^w = (R_{wc}, t_{wc})$ with DROID-SLAM [34], optimizing sparse/semi-dense reprojection together with temporal smoothness:

$$\begin{aligned}\min_{K,\,T_c^w} \sum_{t,i} \left\| \Pi\big(K, (T_c^w)^{-1}; X_i\big) - u_{i,t} \right\|_2^2 \\ + \lambda \Big(\|\Delta t_{wc}\|_2^2 + \left\| \log_{\mathrm{SO}(3)}(R_{wc}^\top R_{wc}^+) \right\|_2^2 \Big),\end{aligned} \tag{6}$$

where $(\cdot)^+$ denotes the next keyframe. We adopt the keyframe solution as the clipwise T_c^w. We also evaluated DPVO and COLMAP but found DROID-SLAM more stable on our setting. With the fixed T_c^w for each clip, we could project our estimated $\mathbf{h}_t, \mathbf{p}_t$ convert to the world frame:

Step 2: Align to canonical action space $\mathcal{A}$. Given the left/right hip and shoulder positions $p_{\text{hip}}^{L/R}$, $p_{\text{sho}}^{L/R}$ in the world coordinate system $\{\mathcal{Z}\}$, we first compute a lateral reference vector on the xz-plane:

$$v_{\text{lat}} \;=\; \Pi_{xz}\big((p_{\text{hip}}^L - p_{\text{hip}}^R) + (p_{\text{sho}}^L - p_{\text{sho}}^R)\big).$$

We then construct the canonical frame $\mathcal{A}$ with the following conventions:

- $z_{\mathcal{A}}$ is aligned with the scene up direction (gravity / ground normal).
- $y_{\mathcal{A}}$ is determined by the dominant interaction direction (e.g., the average hand→object approach vector).
- $x_{\mathcal{A}}$ is obtained by the right-hand rule, ensuring orthonormality.

After orthonormalization, these axes form the rotation matrix $R_{w\to\mathcal{A}} \in \mathrm{SO}(3)$. Finally, we center the action trajectory at the object position (by default, at the first salient frame t_0), giving the translation $t_{w\to\mathcal{A}} = -R_{w\to\mathcal{A}}\, t_o^w(t_0)$ and the resulting rigid transformation $T_w^{\mathcal{A}} = (R_{w\to\mathcal{A}},\, t_{w\to\mathcal{A}})$.

C. Hand Gesture to Robot Arm

C.1. Mapping Algorithm

For the two different gesture types, whole-hand and finger-only, we designed corresponding mapping schemes to translate hand motions to a two-fingered gripper. For whole-hand gestures, our method primarily leverages 3d joints on the palm plane to define the gripper's orientation and spatial position. For finger-only gestures, we incorporate fingertip positions to accommodate dexterous manipulation. The detailed mapping algorithm pseudocode under two gestures is shown in Fig. 10.

Algorithm 1 Whole-Hand (Palm-Involved) Gripper Pose Construction

Input: wrist $\mathbf{k}_{\text{wri}}$, index MCP $\mathbf{k}_{\text{ind}}^{\text{mcp}}$, ring MCP $\mathbf{k}_{\text{ring}}^{\text{mcp}}$

Output: Gripper pose $(R_g, \mathbf{p}_g)$.

$\mathbf{w} \leftarrow \mathbf{k}_{\text{wri}}$, $\mathbf{i} \leftarrow \mathbf{k}_{\text{ind}}^{\text{mcp}}$, $\mathbf{r} \leftarrow \mathbf{k}_{\text{ring}}^{\text{mcp}}$
// Extract keypoints
$\mathbf{o} \leftarrow (\mathbf{w} + \mathbf{i} + \mathbf{r})/3$ *// palm origin*
$v_x \leftarrow (\mathbf{r} - \mathbf{w})$ *// X-axis direction*
$\bar{\mathbf{x}} \leftarrow v_x / (\text{NORMALIZE}(v_x) + 10^{-8})$
// Normalized X-axis
$v_z \leftarrow \text{CROSS\_PRODUCT}(\mathbf{i} - \mathbf{w},\ \mathbf{r} - \mathbf{w})$
// Z-axis (palm normal)
$\bar{\mathbf{z}} \leftarrow v_z / (\text{NORMALIZE}(v_z) + 10^{-8})$
// Normalized Z-axis
$\bar{\mathbf{y}} \leftarrow \text{CROSS\_PRODUCT}(\bar{\mathbf{z}}, \bar{\mathbf{x}})$
// Y-axis direction
$\bar{\mathbf{z}} \leftarrow \text{SIGN}(\diamond) \cdot \bar{\mathbf{z}}$
$R_g \leftarrow \text{CONCATENATE}([\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}}])$
// Rotation matrix
$\mathbf{p}_g \leftarrow \mathbf{o} + d_z\, \bar{\mathbf{z}}$ *// Position*
Return $(R_g, \mathbf{p}_g)$

Algorithm 2 Finger-Only (Pinch/Precision) Gripper Pose Construction

Input: index TIP $\mathbf{k}_{\text{ind}}^{\text{tip}}$, index MCP $\mathbf{k}_{\text{index}}^{\text{mcp}}$, thumb tip $\mathbf{k}_{\text{thumb}}^{\text{tip}}$, thumb MCP $\mathbf{k}_{\text{thumb}}^{\text{mcp}}$

Output: Gripper pose $(R_g, \mathbf{p}_g)$.

$\mathbf{i} \leftarrow \mathbf{k}_{\text{ind}}^{\text{tip}}$, $\mathbf{m} \leftarrow \mathbf{k}_{\text{ind}}^{\text{mcp}}$, $\mathbf{t} \leftarrow \mathbf{k}_{\text{thumb}}^{\text{tip}}$, $\mathbf{r} \leftarrow \mathbf{k}_{\text{thumb}}^{\text{mcp}}$
// Extract keypoints
$\mathbf{o} \leftarrow (\mathbf{t} + \mathbf{i})/2$ *// palm origin*
$v_z \leftarrow (\mathbf{i} - \mathbf{m})$ *// Z-axis direction*
$\bar{\mathbf{z}} \leftarrow v_z / (\text{NORMALIZE}(v_z) + 10^{-8})$
// Normalized Z-axis
$v_y \leftarrow \text{CROSS\_PRODUCT}(\mathbf{i} - \mathbf{m},\ \mathbf{m} - \mathbf{r})$
// Y-axis (palm normal)
$\bar{\mathbf{y}} \leftarrow v_y / (\text{NORMALIZE}(v_y) + 10^{-8})$
// Normalized Y-axis
$\bar{\mathbf{x}} \leftarrow \text{CROSS\_PRODUCT}(\bar{\mathbf{y}}, \bar{\mathbf{z}})$
// X-axis direction
$\bar{\mathbf{z}} \leftarrow \text{SIGN}(\diamond) \cdot \bar{\mathbf{z}}$
$R_g \leftarrow \text{CONCATENATE}([\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}}])$
// Rotation matrix
$\mathbf{p}_g \leftarrow \mathbf{o}$ *// Position*
Return $(R_g, \mathbf{p}_g)$

Figure 10. Following our gesture classification scheme, we divide grasping into two categories and combine them into the final two-finger gripper pose based on the hand joint space orientation.

C.2. Replay Comparison with Different Methods

Replay Comparison by Different Methods: We compared the mapping method of *RoboWheel* with YOTO and GAT-Grasp. YOTO and GAT-Grasp result in discrepancies in gripper position or orientation mapping, leading to failure, while *RoboWheel* provides more accurate and reasonable mapping,as shown in Fig. 11.

C.3. Cross-Arm Retargeting Performance

In Tab. 5, we report the real-world SR of different mapping methods on UR5. To further assess the generalization and scalability of our approach, we deploy the same mapping algorithm across multiple manipulators (UR5, Gen3, iiwa7, Sawyer, and Franka) in simulation. As shown in Tab. 6, our method achieves consistently high performance on all arms, with SR

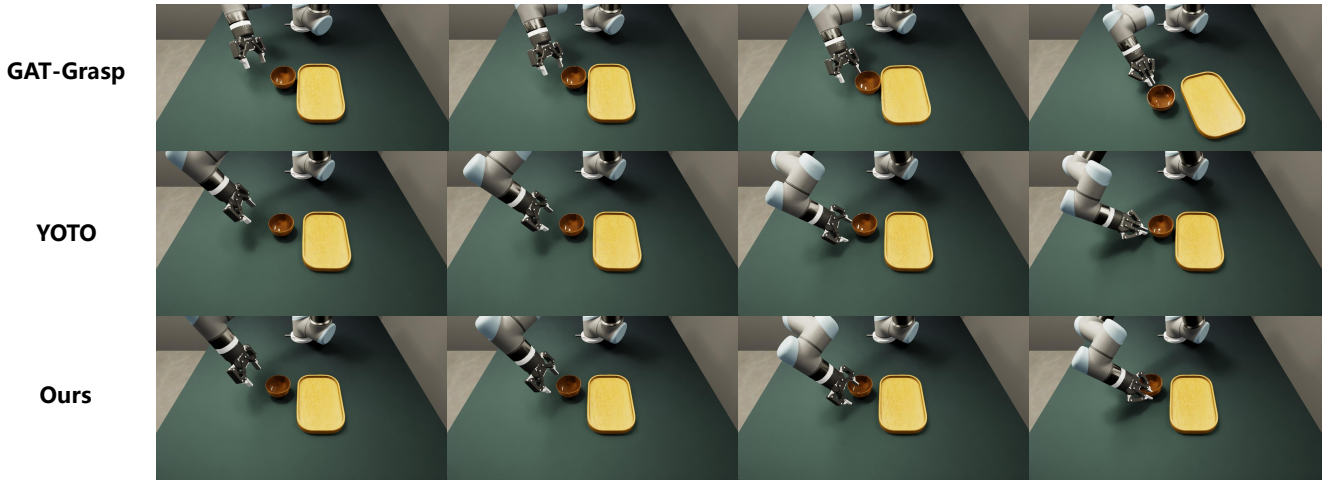

Figure 11. Replay Comparison by Different Methods: We compared the mapping method of *RoboWheel* with YOTO and GAT-Grasp. YOTO and GAT-Grasp result in discrepancies in gripper position or orientation mapping, leading to failure, while *RoboWheel* provides more accurate and reasonable mapping.

Task	**UR5**	**Gen3**	**iiwa7**	**Sawyer**	**Franka**
flip_milk	10/10 (100%)	10/10 (100%)	7/10 (70%)	10/10 (100%)	7/10 (70%)
place_milk	10/10 (100%)	10/10 (100%)	10/10 (100%)	10/10 (100%)	10/10 (100%)
pour_cola	10/10 (100%)	10/10 (100%)	10/10 (100%)	10/10 (100%)	10/10 (100%)

Table 6. Scaled success rates for 10 trials per task using the same retarget method.

remaining stable despite differences in kinematics and embodiments. This indicates that the proposed mapping is robust and scalable for cross-embodiment transfer.

D. Experiment Details

We used the *HORA* to train four VLA/IL policies, namely ACT, DP, RDT-1B, and Pi0, to validate the effectiveness of our data. Before training these models, we preprocessed the data into the format of observations, actions, and instructions (if required) for model training.

D.1. Implementation and Hyper-Parameters of VLA/IL policies

For ACT, we trained each task for 20,000 iterations, with 90% of the data used for training and the remaining 10% for validation. When training DP, we kept the same training steps, learning rate, and chunk size as ACT. The specific hyperparameter values are listed in Tab. 7.

ACT		**DP**	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
Chunk size	16	Chunk size	16
Hidden dim	512	Action horizon	8
Batch size	16	Batch size	16
Learning rate	1e-5	Learning rate	1e-5
Dim feedforward	3200	Observation horizon	8
Training steps	20000	Training steps	20000

Table 7. Hyperparameters used to train ACT and DP.

RDT was pretrained for 100,000 steps with a batch size of 8 per GPU on 4 GPUs, and all single-task fine-tuning was conducted for 10,000 steps with a batch size of 8 per GPU on a single GPUs. **Pi0** was pretrained for 100,000 steps with a batch size of 32 on 8 GPUs, and all fine-tuning was performed for 30,000 steps using the same batch size on a single GPU.

All remaining hyperparameters for RDT and Pi0 were set according to their official documentation.

D.2. Training *HORA* on Maniptrans

We mapped the hand gestures onto the dexterous hands and completed the training in a simulation environment. The training outcomes are shown in Fig. 12.

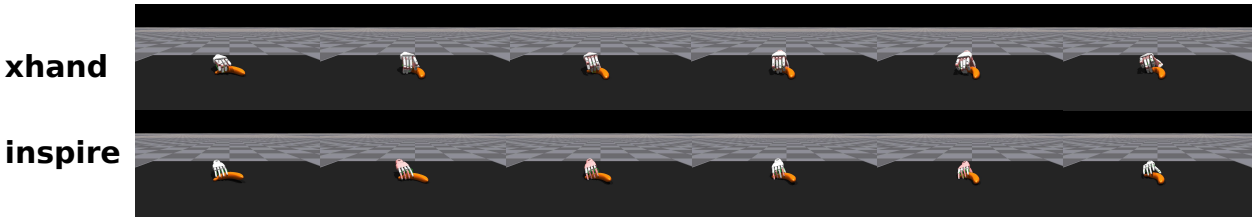

Figure 12. Visualization results of maniptrans

D.3. Real-world Validation Results

Here, we present the experimental results for all designed tasks conducted on physical robot(UR5),as shown in Fig. 13.

Figure 13. Visualization of eight tasks on real robot

E. Dataset Details

We construct a large-scale dataset (*HORA*) with three subsets: *HORA-Mocap* (multi-view mocap with tactile gloves), *HORA-Recordings* (in-house RGB(D) recordings without tactile), and *HORA-Public* (retargeted public HOI datasets).

E.1. *HORA* Mocap Handware Setup

Glove The glove is instrumented with 16 Gen3 tactile sensors and 29 magnetic encoders. Worn on a single hand, it enables the acquisition of high-frequency tactile data. The tactile sensors are capable of detecting pressure, force, and vibration, while the magnetic encoders are used to capture precise joint angles and movements of the fingers. This combination allows for detailed hand-object interaction data to be gathered with high temporal resolution. Visualization of the Mocap Hardware setup and gloves in simulation is shown as Fig. 14.

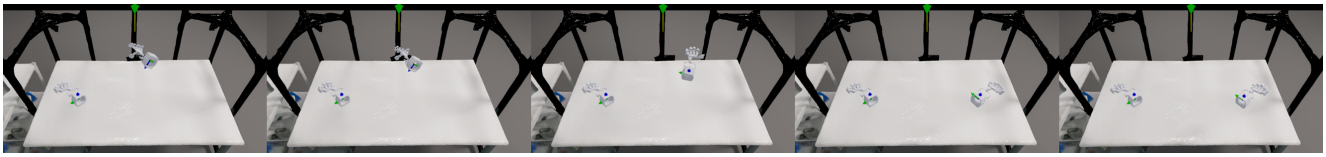

Figure 14. Visualization of the Mocap Hardware setup and gloves in simulation.

RGBD Cameras Three Intel RealSense D455 RGBD cameras are used to capture depth and RGB data simultaneously. The cameras are mounted at strategic locations to ensure optimal coverage and accurate 3D spatial data. The depth cameras provide high-resolution depth maps, while the RGB cameras offer high-quality color images. For synchronized data acquisition, a synchronization cable is employed, connecting three cameras to ensure precise temporal alignment across all devices.

RGB Cameras A total of eight high-resolution RGB cameras are used for detailed visual tracking. These cameras are positioned to cover different angles, enabling comprehensive capture of the environment and subjects. The cameras are used to provide complementary visual data to the depth information provided by the RGBD cameras.

E.2. From Glove Fitting to MANO Parameters

We retarget glove-based kinematic fits to the MANO hand model through a unified differentiable optimization. Both the glove and MANO use differentiable forward kinematics (FK), allowing a smooth transition that preserves global pose and local contact geometry.

Glove-domain fitting. We first calibrate the glove using multi-view constraints. To ensure consistent correspondence, we define: (i) a link-level mapping between glove joints and MANO joints, and (ii) tactile sensor points assigned to glove links. Contact constraints are activated only for links with nonzero force. If a link has no force, its position/normal constraints are disabled. For links under force, we apply force-aware weights: strong-force sensor points have weight 1.0, while other activated points have weight 0.3. The glove fitting objective is

$$\mathcal{L}_{\text{glove}} = \mathcal{L}_{\text{contact}} + \mathcal{L}_{\text{wrist}} + \mathcal{L}_{\text{jlim}} + \mathcal{L}_{\text{smooth}} + \mathcal{L}_{\text{anat}}, \tag{7}$$

where $\mathcal{L}_{\text{contact}}$ aligns positions and normals at tactile points, $\mathcal{L}_{\text{wrist}}$ constrains the wrist pose, $\mathcal{L}_{\text{jlim}}$ penalizes joint-limit violations, $\mathcal{L}_{\text{smooth}}$ enforces temporal smoothness, and $\mathcal{L}_{\text{anat}}$ regularizes anatomical plausibility. All frames are optimized in parallel. To handle wrist floating, we prepend a free 6-DoF transform before the glove wrist base.

Retargeting initialization. The optimized glove motion is used to initialize MANO. We copy the glove global wrist transform (including the 6-DoF floating pose) to MANO's root pose, and map each glove joint rotation to its corresponding MANO joint using the predefined correspondence. This yields an initial MANO pose close to the contact-consistent glove fit.

MANO refinement. Starting from the retargeted initialization, we refine MANO parameters under the same constraints, now expressed on MANO joints/vertices. MANO pose is parameterized by 16 articulated joints, each with 3-DoF axis-angle rotations, i.e.,

$$\boldsymbol{\theta} \in \mathbb{R}^{16\times 3}, \tag{8}$$

together with a global wrist pose. We minimize

$$\mathcal{L}_{\text{mano}} = \mathcal{L}_{\text{contact}} + \mathcal{L}_{\text{wrist}} + \mathcal{L}_{\text{jlim}} + \mathcal{L}_{\text{smooth}} + \mathcal{L}_{\text{anat}}, \tag{9}$$

using the same force-aware contact weights transferred from glove links. This refinement produces anatomically valid MANO poses while preserving tactile interaction geometry.

E.3. Task Description for *HORA*

The *HORA*-Mocap subset is designed to provide a compact yet diverse set of everyday household manipulation skills captured with high-fidelity motion tracking. We curate tasks that range from simple, atomic primitives (e.g., pick-and-place, pressing, pouring, inserting, and opening/closing) to mid- and long-horizon activities that require multi-step coordination and object-centric reasoning (e.g., pouring water into a cup, storing blocks, tabletop cleaning, tong-based picking, and bulb installation). To reflect realistic human hand usage, the task taxonomy is simplified into one-handed and two-handed categories, covering both single-handed interactions and cooperative bimanual behaviors such as rotating a cap, folding clothes, scanning items, and table setting. For detailed task descriptions and visualization, please refer to the Tab.8 and Fig.15. The resulting task suite offers a structured benchmark for learning and evaluating dexterous manipulation policies under varied object types, action primitives, and temporal horizons.

#	Task type	Primitive / skill	Manipulated objects	Task description
1	One-handed	pick&place	Coconut water bottle (350 ml)	Pick up and place the object.
2	One-handed	pick&place	Black marker	Pick up and place the object.
3	One-handed	pick&place	Lidded ceramic mug (white)	Pick up and place the object.
4	One-handed	pick&place	Claw hammer	Pick up and place the object.
5	One-handed	press	Remote control	Press buttons on the remote control.
6	One-handed	pour	Kettle	Pick up the kettle and perform a pouring motion.
7	One-handed	insert	Gel pen + pen holder	Insert the pen into the pen holder.
8	One-handed	open–close	Tea gift box	Open the tea gift box and then close it.
9	One-handed	pouring water	Kettle + cup	Place the cup at a fixed position, pick up the kettle to pour water, put down the kettle, then deliver the cup to a target position.
10	One-handed	block storage	Blocks + block box	Store blocks of different shapes into the block box.
11	One-handed	tabletop wiping	Cleaning sponge	Hold the sponge and wipe the tabletop.
12	One-handed	vacuum cleaning	Handheld vacuum	Use the handheld vacuum to clean the tabletop.
13	One-handed	tong picking	Baking tongs + bun-shaped object	Use tongs to pick up the bun-shaped object.
14	One-handed	bulb installation	Light bulb + bulb socket	Screw the light bulb into the socket.
15	Two-handed	pick&place	Large plate	Pick up and place the object with both hands.
16	Two-handed	pick&place	Large juice bottle	Pick up and place the object with both hands.
17	Two-handed	rotate	Bottle with cap	Cooperatively unscrew the cap with both hands, then re-tighten it.
18	Two-handed	clothes folding	T-shirt	Fold the T-shirt and place it at a fixed position.
19	Two-handed	item scanning	Scanner gun + items to scan	Pick up an item and scan it with the scanner gun.
20	Two-handed	table setting	Tableware set	Set the table following Western-style table-setting rules.

Table 8. *HORA*-Mocap subset task list.

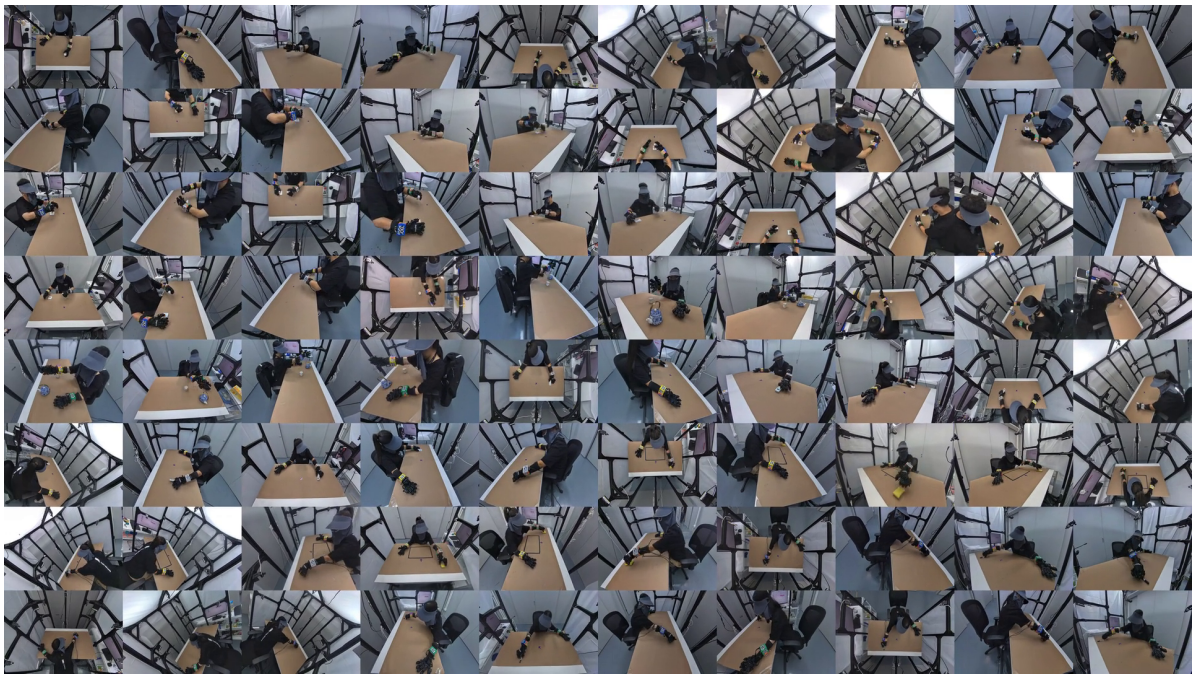

Figure 15. *HORA*-Mocap Overview,

F. Data Augmentation

In this section, we present various data augmentation strategies and additional details applied to more tasks.

F.1. Cross-Embodiment

Here, we present additional results from the data augmentation module, demonstrating motion retargeting for HOI reconstruction to different robotic arms. Fig. 16 and Fig. 17 show the retargeted motions for the tasks "flip milk," "pour water," and "place milk" to the *UR5/UR5e*, *Franka Emika Panda*, *KUKA LBR iiwa 7*, *Kinova Gen3*, and *Rethink Robotics Sawyer* arms, respectively.

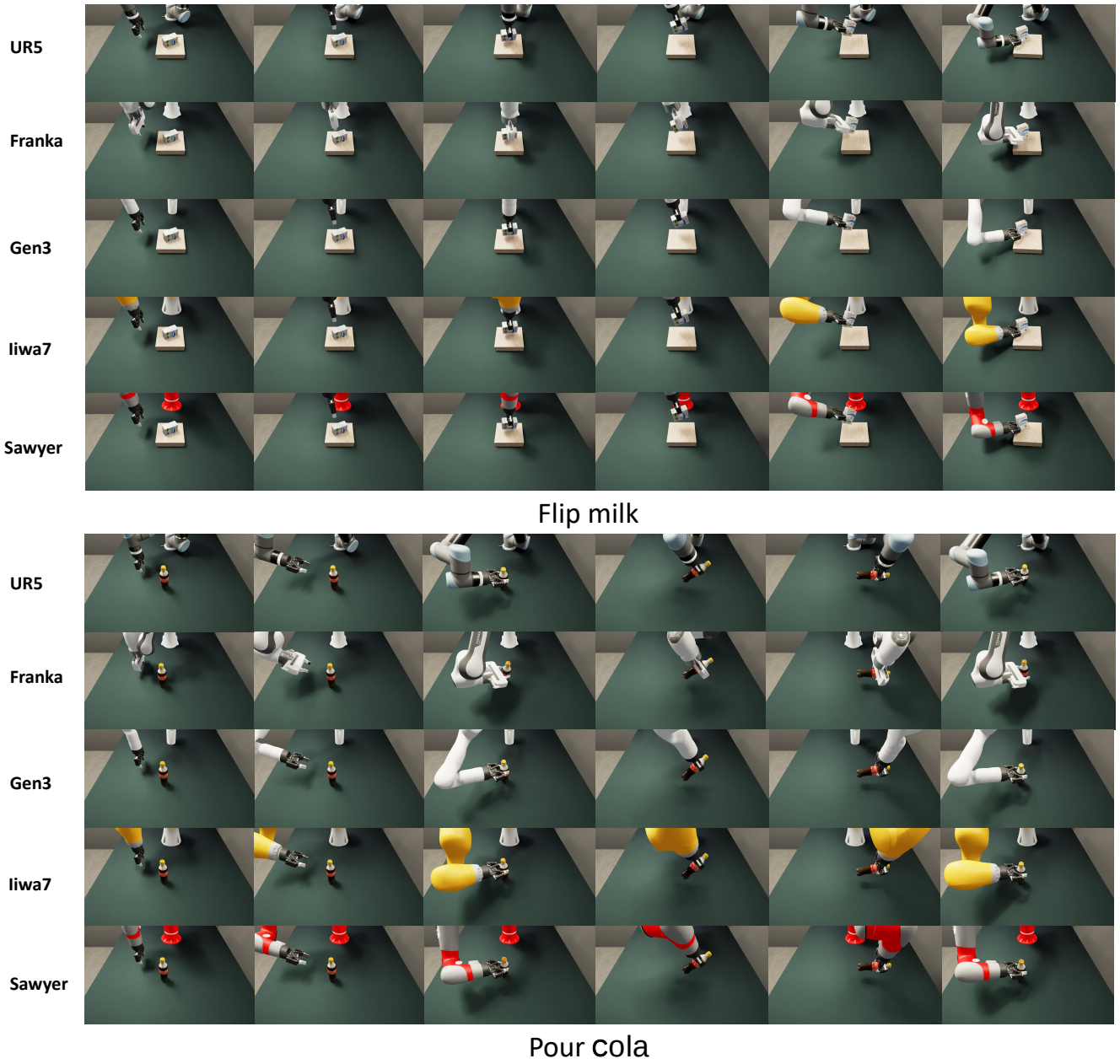

Figure 16. Visualization of robot arm augmentation:flip milk and pour Water

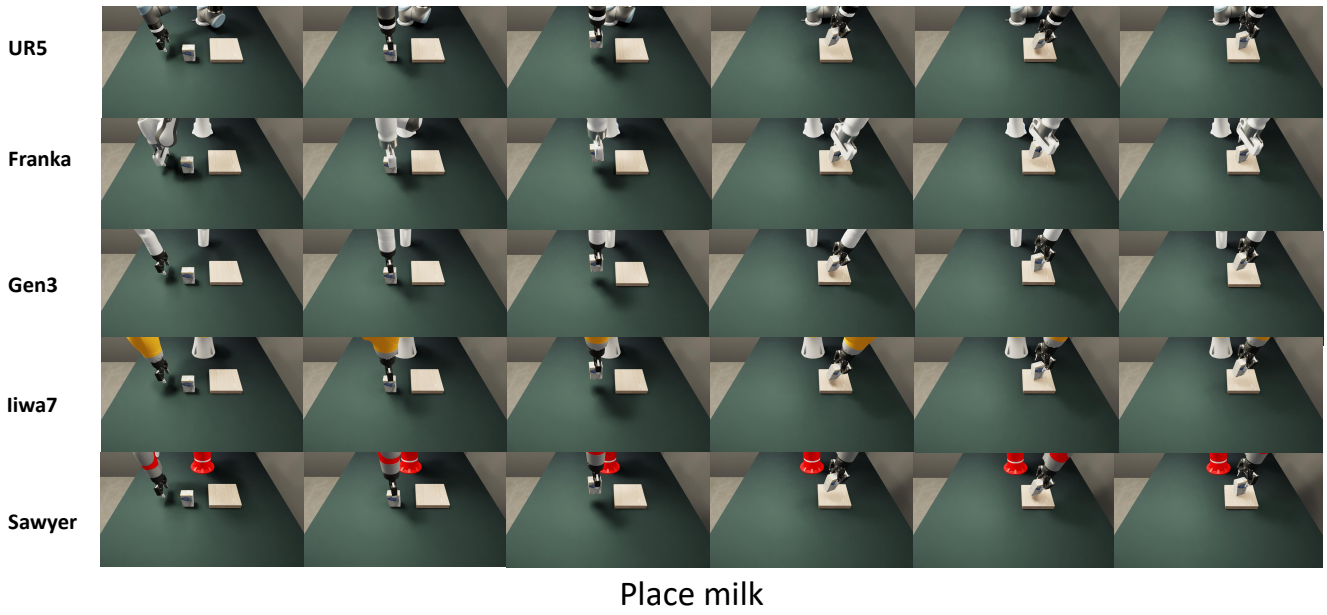

Figure 17. Visualization of robot arm augmentation:place milk

F.2. Background Variation

As shown in Fig. 18, we apply scene-level visual randomization to diversify the pixel distribution while keeping task dynamics and contact semantics unchanged: *i*) workspace and background appearance randomization (e.g., tabletop, backsplash) via texture and normal-map swaps, and adjustment of basic PBR parameters (albedo/roughness); *ii*) illumination randomized using parametric light sources with variations in spatial placement, intensity, color, color temperature, and emission radius, enabling a broad range of plausible lighting conditions; *iii*) clutter regime ranging from empty scenes to heavy distractors, with randomly sampled object positions and orientations placed collision-free outside the robot's swept volume via rejection sampling; *iv*) mild camera intrinsics/extrinsics jitter consistent with prior calibration to emulate plausible view changes.

F.3. Object Retrival

Our object-retrieval augmentation strategy successfully enables the transfer of manipulation skills to novel objects in simulation. By replacing the original object with a retrieved counterpart that shares high geometric and semantic similarity, and initializing it in the same canonical pose, the robot can reliably execute the same action trajectory. Visually confirmed in Fig. 19, Fig. 20, and Fig. 21 for tasks including "pour water", "tip tea cup", and "place box".

F.4. Hand Mirror

Motivation. Many daily manipulations are left–right symmetric up to a sagittal-plane reflection; mirroring increases trajectory diversity without changing task semantics.

Operator. Let the sagittal reflection be $S = \mathrm{diag}(-1, 1, 1)$. For positions, $p'(t) = S\,p(t)$. A pure reflection is improper for orientations, so we compose a π-rotation about the y-axis to recover a proper rotation:

$$R'(t) \;=\; S\,R(t)\,S \cdot R_y(\pi), \qquad \det\big(R'(t)\big) = +1. \tag{10}$$

We mirror *both* hand and object about the same plane so that $T'_{\mathrm{rel}}(t) = T'_h(t)^{-1} T'_o(t) = T_{\mathrm{rel}}(t)$, preserving contact frames and approach vectors. Gripper chirality and finger-axis signs are flipped consistently.

Safeguards. We exclude actions whose handedness encodes semantics (e.g., threaded fasteners), detected via a non-zero screw component about the task z-axis exceeding τ_{screw}. Mirrored rollouts must pass the replay check on a reference arm before inclusion.

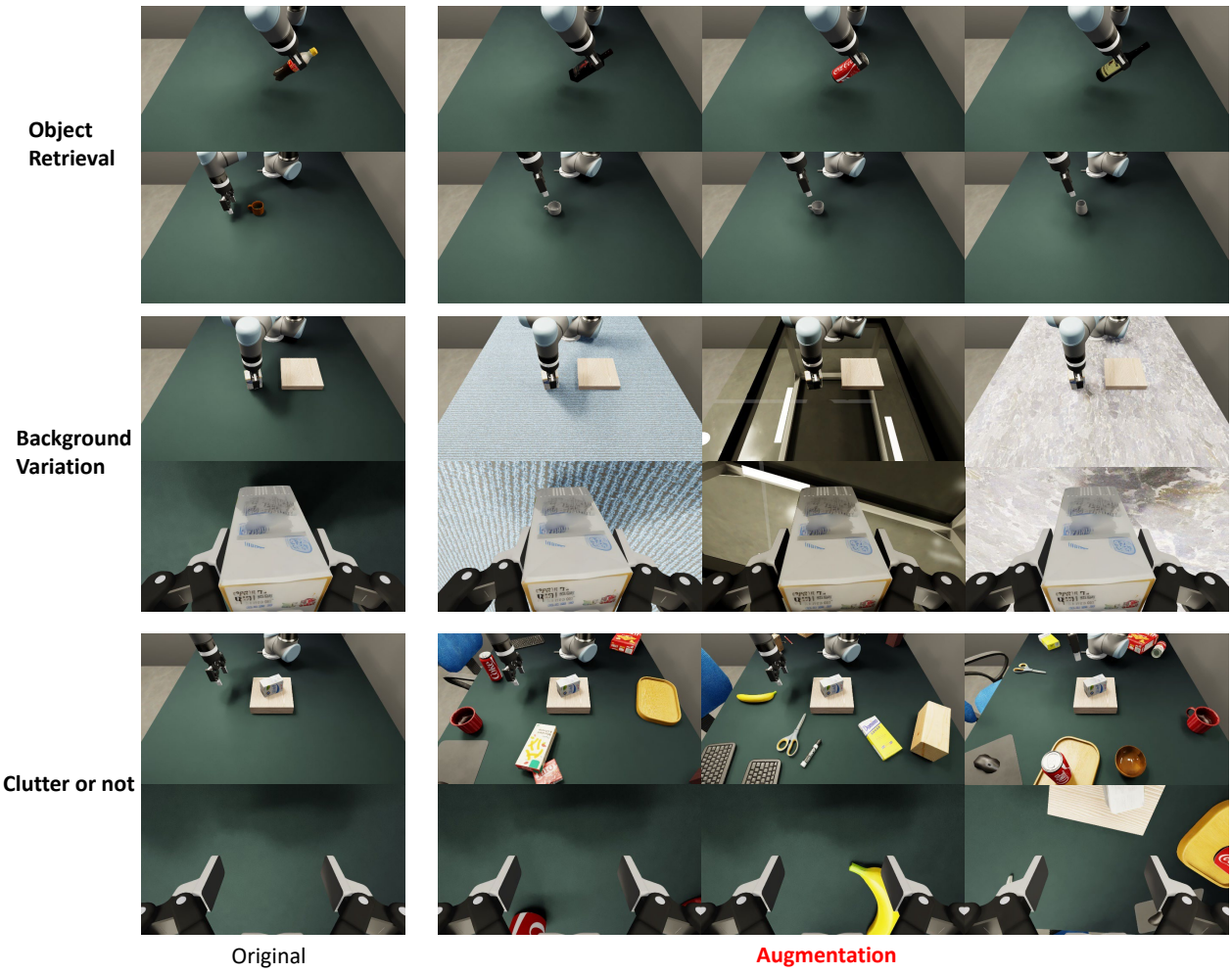

Figure 18. Diverse background texture augmentation in *RoboWheel*.

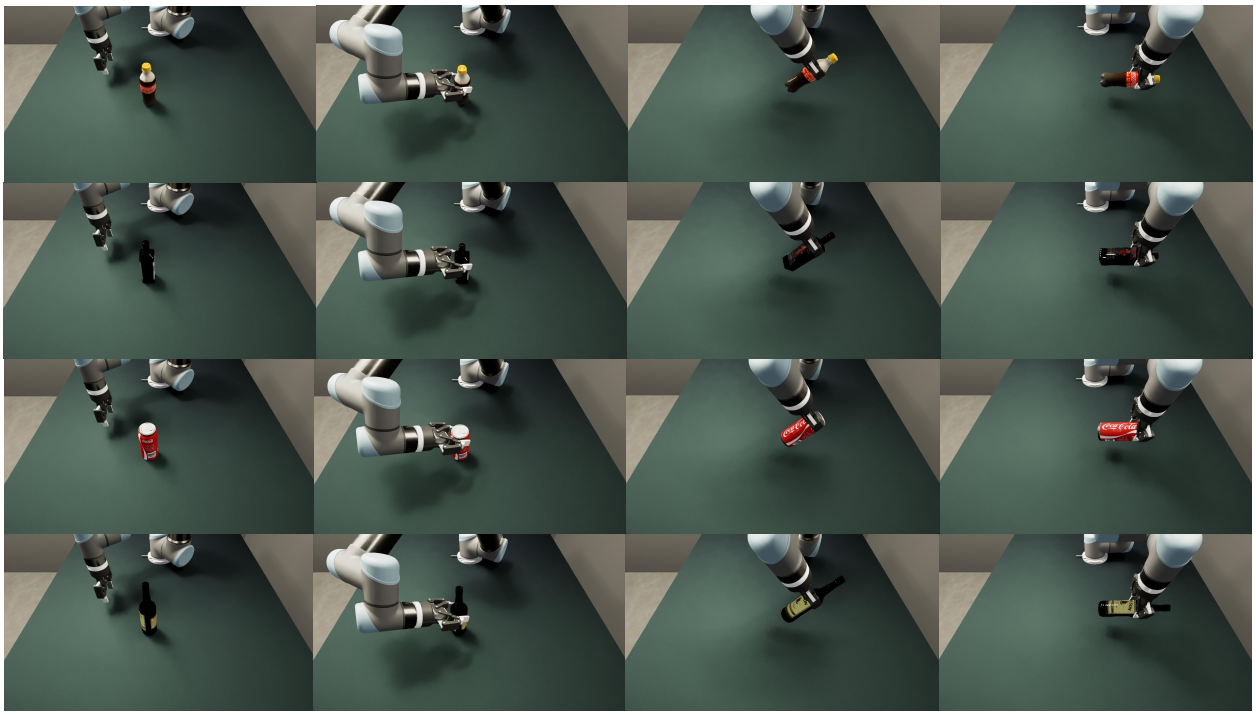

Figure 19. Object Retrieval augmentation

Figure 20. Object Retrieval augmentation

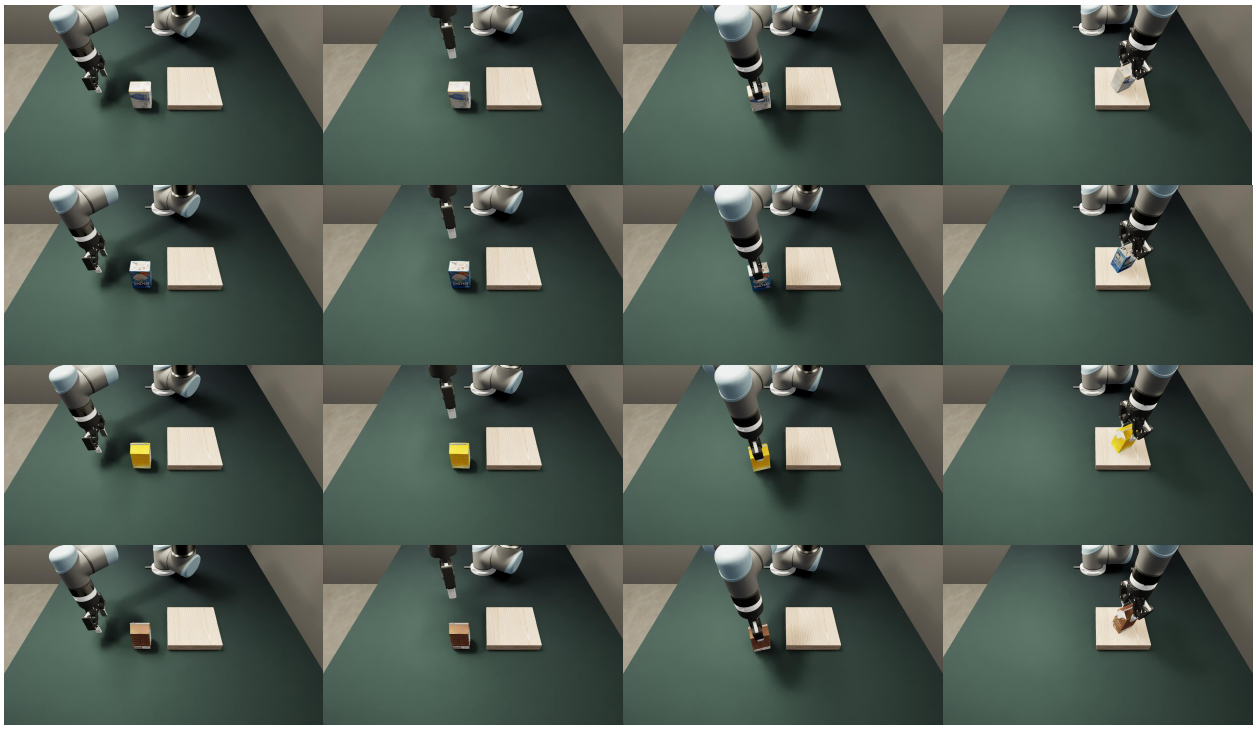

Figure 21. Object Retrieval augmentation