

Teleoperation, Simulation, or Human Video? Data Utilization Law for Robot Manipulation

Supplementary Material

7. Experimental Environment and Implementation Details

7.1. Hardware Setup

Our experiments were conducted on the following hardware and software platforms.

- **Computational Platform:** All models were trained and evaluated on a server equipped with $8 \times$ NVIDIA H20 GPUs
- **Teleoperation Platforms:** We utilized two distinct bimanual robot setups:
 - **Aloha-Agilex-2.0:** This platform is equipped with three ORBBEC DaBai cameras, providing top-down, left-wrist, and right-wrist views.
 - **Dual-ARX-R5:** This setup uses an Intel RealSense D457 camera for the top-down view and two Intel RealSense D435 cameras for the wrist-mounted views.
- **Human Video Capture:** Human demonstration videos were recorded using a DJI Action Pro 5 camera at a resolution of 1080×960 pixels and a frame rate of 30 Hz.

7.2. Detailed Manipulation Task Definitions

We define three bimanual manipulation tasks, each with specific initial conditions, goal states, and success criteria.

- **Pick Dual Bottles:** The task requires the robot to pick up two bottles (e.g., a Coca-Cola and a Sprite bottle, each with specific dimensions and weight) from randomized initial positions within designated zones. The goal is to place them into their respective target regions. The task is scored cumulatively:
 - 0.5 points for successfully grasping the Sprite bottle with the right arm.
 - An additional 0.5 points for grasping the Coca-Cola bottle with the left arm.
 - A full score of 1.0 is awarded only when both bottles are lifted simultaneously to a target height.
- **Hand Over Block:** The agent must first use its left arm to pick up a red block from a specified area (0.5 points). It must then perform a dexterous hand-over, passing the block to the right arm. A full score of 1.0 is awarded upon the right arm successfully placing the block into a designated blue goal region in the top-right corner of the workspace.
- **Rank RGB Blocks:** The workspace contains three blocks of distinct colors: Red (R), Green (G), and Blue (B). Their initial arrangement is randomized among three permutations: (G, R, B), (G, B, R), and (R, B, G). The goal is to

use both arms to rearrange the blocks into a specific target sequence (R, G, B) within a designated area. Success is measured by the number of correctly placed blocks, with 0.33 points awarded for each, up to a maximum of 1.0 point.

7.3. Optimal Allocation Under a Time Budget

We formulate the allocation problem as a *cost-benefit knapsack*, aiming to maximize the number of *real-equivalent trajectories* within a fixed time budget B . The time cost consists of both real-world and simulated data collection: each real trajectory takes C_{real} hours, and each simulated trajectory takes C_{sim} hours. Since simulated data contributes less to task performance, we introduce a conversion factor η , representing the number of simulated trajectories equivalent to a single real trajectory.

To determine the optimal allocation, we define the cost-effectiveness ratio of simulated data as:

$$\rho = \frac{C_{\text{sim}}}{C_{\text{real}}} \times \eta.$$

When $\rho < 1$, simulated data is more cost-effective after conversion, and both types of data should be used. When $\rho \geq 1$, allocating all available time to real data yields the best result. The equivalent threshold condition is:

$$\eta < \frac{C_{\text{real}}}{C_{\text{sim}}}.$$

For the mixed allocation case ($\eta < \frac{C_{\text{real}}}{C_{\text{sim}}}$), the optimal numbers of simulated and real trajectories are computed as:

$$N_{\text{sim}}^* = \text{int} \left(\frac{B}{C_{\text{sim}} + \frac{C_{\text{real}}}{\eta}} \right),$$
$$N_{\text{real}}^* = \text{int} \left(\frac{B - C_{\text{sim}} \cdot N_{\text{sim}}^*}{C_{\text{real}}} \right),$$

where $\text{int}(\cdot)$ denotes rounding down to the nearest integer. This allocation ensures the total time consumption does not exceed B and maximizes the number of real-equivalent trajectories.

8. Dataset Details

This section provides a comprehensive overview of the data collection, processing, and statistics for the three data sources used in our work: Teleoperation demonstrations, simulated data, and human videos.

8.1. Teleoperation Data Collection

- **Teleoperation Process:** Expert demonstrations were collected via teleoperation. For the Aloha-Agilex-2.0, we used a symmetric leader-follower setup controlled via the `cobotmagic` interface. For the Dual-ARX-R5, an expert operated the arms using a Virtual Reality (VR) interface. Data from both platforms was recorded at 30 Hz to ensure high-fidelity motion capture.
- **Data Format:** Each timestamped data point comprises three synchronized RGB image streams (640×480 resolution) from the top, left-wrist, and right-wrist cameras, paired with the corresponding 14-dimensional robot action vector in joint space.

A summary of the collected teleoperation demonstration data is provided in Table 4.

8.2. Simulation Data Generation

- **Simulation Environment:** We generated a large-scale synthetic dataset using our *RoboTwin 2.0* simulation environment, which is built on the [Specify Engine, e.g., MuJoCo or Isaac Gym] physics engine. Physics parameters, such as friction coefficients and gravitational constants, were carefully calibrated to closely mirror real-world dynamics.
- **Domain Randomization:** To foster the learning of robust representations, we applied extensive domain randomization across several axes:
 - **Background Textures:** Tabletop and background wall textures were randomly sampled from a large library, including procedurally generated patterns and images from public datasets [Specify source, e.g., Poly Haven].
 - **Lighting Conditions:** The position, color, and intensity of multiple light sources in the scene were randomized for each episode.
 - **Distractor Objects:** To simulate a cluttered environment, we randomly placed a variety of distractor objects (with randomized shapes, sizes, and colors) on the tabletop, avoiding the primary task interaction areas.

The statistics for the generated simulation data are summarized in Table 5.

8.3. Human Video Data Collection and Processing

Collection Protocol. Human demonstrators performed the three manipulation tasks on a white tabletop that matched the robot’s workspace in layout and object configuration. A DJI Action Pro 5 camera was chest-mounted on the operator to capture a first-person-like egocentric viewpoint at 1080×960 resolution and 30 Hz. Demonstrators were instructed to perform actions clearly and to minimize hand-object occlusions. We collected 500 trajectories per task, totaling 1,500 demonstrations.

Processing Pipeline. Raw video footage is processed through the following stages:

1. **Video Stabilization.** We apply Lucas-Kanade sparse optical flow [19] to estimate frame-to-frame affine transforms (translation and rotation). The estimated camera trajectory is smoothed with a uniform filter (window size 30 frames), and corrective affine warps are applied to each frame to remove residual camera shake.
2. **Hand Landmark Extraction.** We run MediaPipe Hand Landmarker [20] in video mode to detect and track both hands, extracting 21 3D landmarks per hand per frame in world coordinates.
3. **End-Effector Pose Computation.** For each hand, we compute a 6D end-effector representation: (i) a 3D wrist position (landmark 0 in world coordinates); (ii) a 3×3 rotation matrix constructed from three landmarks — the Z-axis points from the wrist toward the middle-finger MCP (landmark 9), the X-axis is the cross product of Z and the wrist-to-ring-finger-MCP direction, and the Y-axis completes the right-handed frame.
4. **Gripper State Estimation.** The gripper state $g \in [0, 1]$ is estimated from the Euclidean distance between the thumb tip (landmark 4) and index fingertip (landmark 8) in world coordinates. Distances below a threshold $\tau = 0.03$ m map to $g = 1.0$ (closed); distances above 2τ map to $g = 0.0$ (open); values in between are linearly interpolated.
5. **Action Vector Construction.** Frame-to-frame deltas are computed: position delta $\Delta p = p_t - p_{t-1}$ and rotation delta $\Delta r = \log(R_t R_{t-1}^\top)$ (rotation vector, 3D). The final per-frame action is a **14-dimensional** bimanual vector:

$$a = [\Delta p_L, \Delta r_L, g_L, \Delta p_R, \Delta r_R, g_R] \in \mathbb{R}^{14}.$$

6. **Storage.** Each processed trajectory is stored in HDF5 format. Each file contains the action sequence, end-effector poses, and synchronized RGB frames resized to 640×480 .

9. Policy Training

9.1. Model Architectures

Base Backbones. We employ two distinct policy backbones to examine architecture-agnostic generalization. Our primary backbone is the Diffusion Policy [8], implemented using a CNN-based U-Net architecture. Visual observations are processed via a ResNet-50 encoder pretrained on ImageNet, producing spatial feature maps that are projected into the latent space of the U-Net. Language-conditioned inputs are embedded using DistilBERT [27], whose outputs are concatenated with visual latents prior to the denoising stage.

The secondary backbone, denoted as π_0 , is initialized with PaliGemma pre-trained weights [29]. Unlike Diffu-

Table 4. Statistics of the collected Teleoperation demonstration dataset.

Platform	Task	# Trajectories	Avg. Length (Steps)	Total Steps
Aloha-Agilex-2.0	Pick Dual Bottles	200	450	90,000
	Hand Over Block	300	300	90,000
	Rank RGB Blocks	500	600	300,000
Dual-ARX-R5	Hand Over Block	200	350	70,000
	Rank RGB Blocks	300	650	195,000

Table 5. Statistics of the generated simulation dataset.

Task	# Trajectories	Avg. Length (Steps)	Total Steps
Pick Dual Bottles	10,00	450	4,500,00
Hand Over Block	20,00	300	6,000,00
Rank RGB Blocks	20,00	600	12,000,00

sion Policy’s explicit multimodal fusion layers, π_0 adopts a transformer-based joint vision-language encoder, enabling native handling of mixed-modality inputs through shared cross-attention mechanisms.

Human-Video Pretraining. For cross-domain transfer, both models are initially pre-trained on the human demonstration dataset (*human video*), where action trajectories are represented as 14-dimensional end-effector delta vectors encoding bimanual wrist position deltas, rotation deltas, and gripper states. Although this representation differs semantically from the robot’s joint-angle action space used during fine-tuning, this cross-representation transfer is by design. Following the philosophy of H-RDT [2], pre-training on human video primarily develops manipulation-aware visual representations and motion priors in the shared policy backbone, rather than learning a direct action-space correspondence. During fine-tuning on teleoperation data, the policy re-adapts its action outputs to the robot’s joint-angle space while retaining the transferred visual understanding. The visual encoder and backbone weights are directly transferred to the fine-tuning stage for both Diffusion Policy and π_0 .

9.2. Training Hyperparameters

Table 6. Training Hyperparameters for Diffusion Policy and π_0 .

Hyperparameter	Diffusion Policy	π_0
Optimizer	AdamW	AdamW
Learning Rate	1×10^{-4}	2×10^{-5}
Weight Decay	0.01	0.01
Batch Size	768	64
Total Training Steps (T_{total})	13000	40000
Pre-training Steps (T_{pre})	10,000	30,000
Fine-tuning Steps (T_{ft})	3000	10,000