

## 524 A Task Details

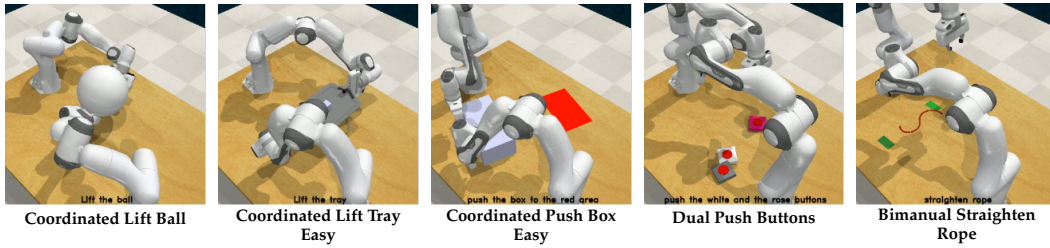


Figure 6: Simulation environments for our bimanual manipulation tasks, adapted from PerAct2 [75].

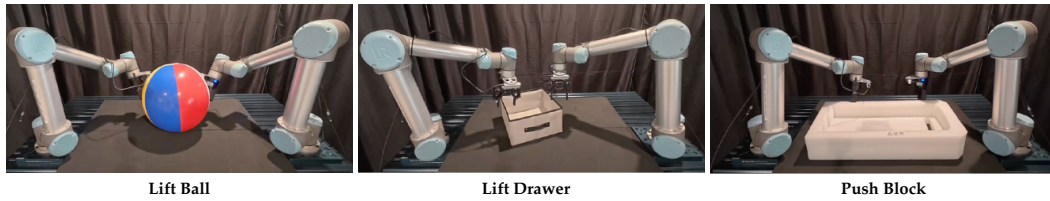


Figure 7: Real-world bimanual manipulation tasks.

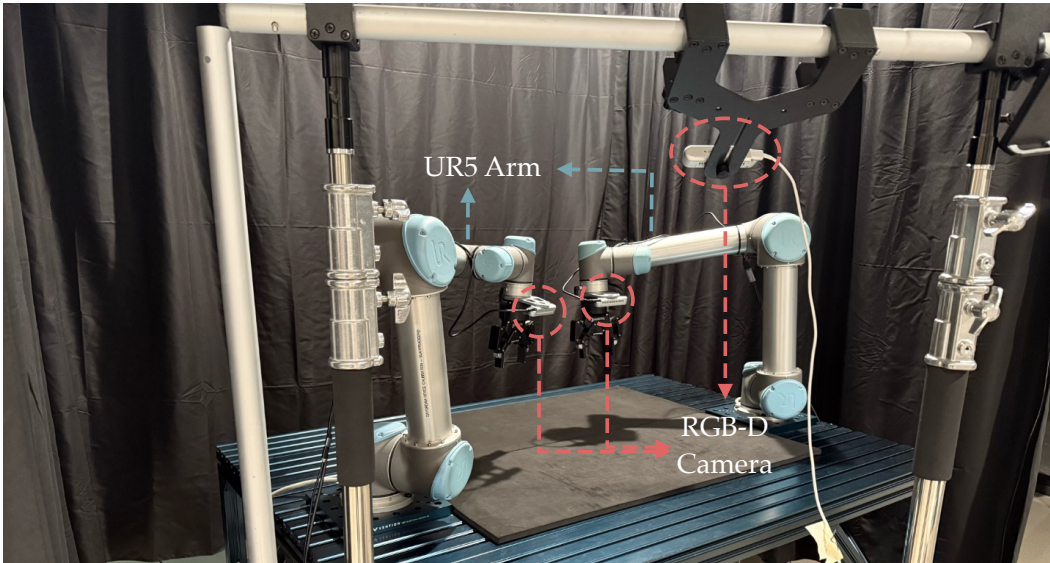


Figure 8: Real-world bimanual UR5 setup.

## 525 B Ablations

526 **Ablations of D-CODA.** In simulation, we test the following methods:

- 527 • **D-CODA with Replaced Encoders:** uses a VQGAN encoder trained on the Open Images [79]
- 528 dataset from Latent Diffusion [71] instead of the RealEstate10K [80] dataset.
- 529 • **D-CODA w/o Constrained Optim.:** does not use constrained optimization to sample camera
- 530 perturbations (i.e., random sampling).

531 Table 3 indicates that D-CODA performs best with  
 532 constrained optimization and the original VQGAN  
 533 encoder. D-CODA without constrained optimiza-  
 534 tion fails more often than D-CODA during ball  
 535 lifting, as expected, since the perturbations gener-  
 536 ated by the model are not constraint-enforced and

| Method                        | Coordinated Lift Ball |
|-------------------------------|-----------------------|
| D-CODA with Replaced Encoders | 53.3                  |
| D-CODA w/o Constrained Optim. | 57.3                  |
| <b>D-CODA (ours)</b>          | <b>73.3</b>           |

Table 3: Ablation experiment results in simulation.

are entirely random. See Section G for details on the importance of constraint-enforced action sampling. Additionally, the model with replaced encoders generates images with more artifacts, resulting in poorer policy performance.

## C Real-World $\pi_0$ -FAST Experiment

We fine-tune a vision-language-action model,  $\pi_0$ -FAST [81], on the original and augmented data described in Section 5.  $\pi_0$ -FAST (w/o augment.) is trained only on the original data, whereas D-CODA (ours) is trained on both the original and augmented data. Fine-tuning is performed using Low-Rank Adaptation (LoRA) with the Gemma-2B-LoRA variant for 150,000 training steps, provided in [81]. In Lift Ball, the baseline frequently misses the ball by moving the arms over it, or squeezes the ball so tightly that it triggers a force limit error on the robot. In contrast, D-CODA more reliably completes the task by positioning the arms beneath the ball to lift it, a strategy that is not observed in the baseline executions. However, most failures of D-CODA are due to large action values generated from the policy, causing the arms to deviate from the intended trajectory. We suspect that the large actions may result from the discontinuous nature of action tokens, as the augmented states are out-of-distribution, perturbed original states, which could inadvertently cause the policy to learn to output actions that suddenly deviate from the trajectory. This issue does not appear in ACT and might be mitigated by adopting a smoother action token representation. In Lift Drawer, both D-CODA and the baseline achieve the same success rate. However, in 12 out of 20 trials, D-CODA has at least one gripper reach the side of the drawer (an intermediate subgoal necessary to complete the task), compared to only 9 out of 20 trials for the baseline. In Push Block, both D-CODA and the baseline succeed in 20 out of 20 trials. Overall, both  $\pi_0$ -FAST and ACT demonstrate improved performance with our data augmentation; however, with limited training data, ACT appears to exhibit greater reliability.

| Method                       | Lift Ball | Lift Drawer | Push Block |
|------------------------------|-----------|-------------|------------|
| $\pi_0$ -FAST (w/o augment.) | 2 / 20    | 1 / 20      | 20 / 20    |
| D-CODA (ours)                | 12 / 20   | 1 / 20      | 20 / 20    |

Table 4: Real-world experiment results comparing  $\pi_0$ -FAST without data augmentation and  $\pi_0$ -FAST with D-CODA, with 20 trials per method and task combination.

## D Generalization Experiment

We evaluate the diffusion model’s generalization capability to unseen objects and tasks. For the zero-shot and few-shot experiments, we train the model on 100 demonstrations each from the following PerAct2 [75] tasks: Coordinated Lift Tray, Pick Up Notebook, Pick Up Plate, Sweep Dust Pan, and Coordinated Push Box. We then use the trained model to synthesize images for the Coordinated Lift Ball dataset. The following methods are tested:

| Method                         | Coordinated Lift Ball |
|--------------------------------|-----------------------|
| Zero-Shot                      | 44.0                  |
| Few-Shot (10 demos)            | 60.0                  |
| Train from Scratch (100 demos) | 73.3                  |

Table 5: Ablation experiment results in simulation.

- **Zero-Shot:** uses the trained diffusion model to synthesize images without any fine-tuning.
- **Few-Shot (10 demos):** fine-tunes the trained model for 3000 additional epochs using 10 demonstrations from the target Coordinated Lift Ball dataset, which is then use for image synthesis.
- **Train from Scratch (100 demos):** Trains the diffusion model directly on 100 demonstrations from the Coordinated Lift Ball dataset, without using demonstrations from other tasks.

As shown in Table 5, the diffusion model performs best when trained directly on the target dataset (i.e., the dataset to be augmented). However, when data collection for the target task is costly, the model still achieves reasonable performance in the few-shot setting. Qualitatively, the images synthesized by the model trained from scratch contain the fewest artifacts, with image quality degrading as fewer target demonstrations are used during training.

## E Additional Implementation Details

For training the diffusion model, we use the same VQ-GAN pre-trained checkpoint at 2000 epochs as DMD [9]. To randomly sample images  $I_a^l, I_b^l, I_a^r, I_b^r$  from a robot trajectory to construct the input  $(I_a^l, I_b^l, \Delta p^l, I_a^r, I_b^r, \Delta p^r)$ , we sample from a range of  $\{5, \dots, 15\}$  for all simulation tasks, except for Dual Push Buttons, where we use  $\{10, \dots, 30\}$ . For real-world experiments, we use a range of  $\{1, \dots, 3\}$  for all tasks. For example, if  $I_a^l, I_a^r$  are from timestep  $t$ , then  $I_b^l, I_b^r$  are sampled from a future timestep between  $t + 1$  and  $t + 3$  in real-world tasks. We use two NVIDIA 4060 Ti GPUs for both training the diffusion model and performing image synthesis.

For camera perturbation sampling, the translation magnitudes  $[m_{lb}, m_{ub}]$  are set to 0.01 and 0.02 meters, respectively, for contactless and contact-rich states. For contactless states, the rotation bounds  $[r_{lb}, r_{ub}]$  are set to  $-28.7$  and  $28.7$  degrees, respectively. For  $k$  (i.e., the interval at which original states are replaced), we set  $k = 6$  for Coordinated Lift Ball, Coordinated Lift Tray Easy, Dual Push Buttons, and all real-world tasks, and  $k = 9$  for Coordinated Push Box Easy and Bimanual Straighten Rope. Figure 9 shows that our method is largely insensitive to the choice of  $k$ , which motivates our choice of 6 as the default value for most tasks.

Table 6 summarizes the ACT hyperparameters. While PerAct2 uses a default action chunk size of 10, we found it to yield suboptimal performance across most tasks. To address this, we tune the chunk size for all tasks except Coordinated Lift Ball, using a chunk size of 15 for Coordinated Lift Tray Easy and Coordinated Push Box Easy, and 60 for Dual Push Buttons and Bimanual Straighten Rope. We use a chunk size of 2 across all real-world tasks. In both simulation and real-world experiments, the RGB images have dimensions of  $128 \times 128$ . An NVIDIA 2080 Ti GPU is used to train the ACT policy.

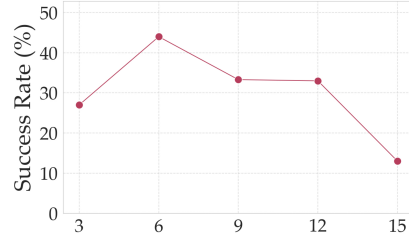


Figure 9: Effects of  $k$  on downstream ACT performance on Coordinated Lift Tray Easy.

| Hyperparameter        | Value |
|-----------------------|-------|
| learning rate         | 1e-5  |
| batch size            | 16    |
| # encoder layers      | 4     |
| # decoder layers      | 7     |
| feedforward dimension | 3200  |
| hidden dimension      | 512   |
| # heads               | 8     |
| beta                  | 100   |
| dropout               | 0.1   |

Table 6: Hyperparameters of ACT

For real-world experiments, we use Intel RealSense D415 cameras to capture RGB images at a resolution of  $640 \times 480$  pixels. These images are first zero-padded and then rescaled to  $128 \times 128$ . We use the [python-urx](#) library to control the robot arms and I/O programming to operate the Robotiq 2F-85 grippers.

## F Additional Implementation Details for the Baselines

For fine-tuned VISTA, 10 overhead camera viewpoints are randomly sampled from a quarter-circle arc distribution and are used to train ZeroNVS with VISTA’s default fine-tuning parameters. The ZeroNVS model is fine-tuned for 5,000 steps on four NVIDIA A40 GPUs. The resulting model is then used to synthesize overhead camera views for all timesteps in each episode. These synthesized images replace all the original overhead images and are used to train ACT.

622 **G Lack of Constraint-Enforced Action Sampling**

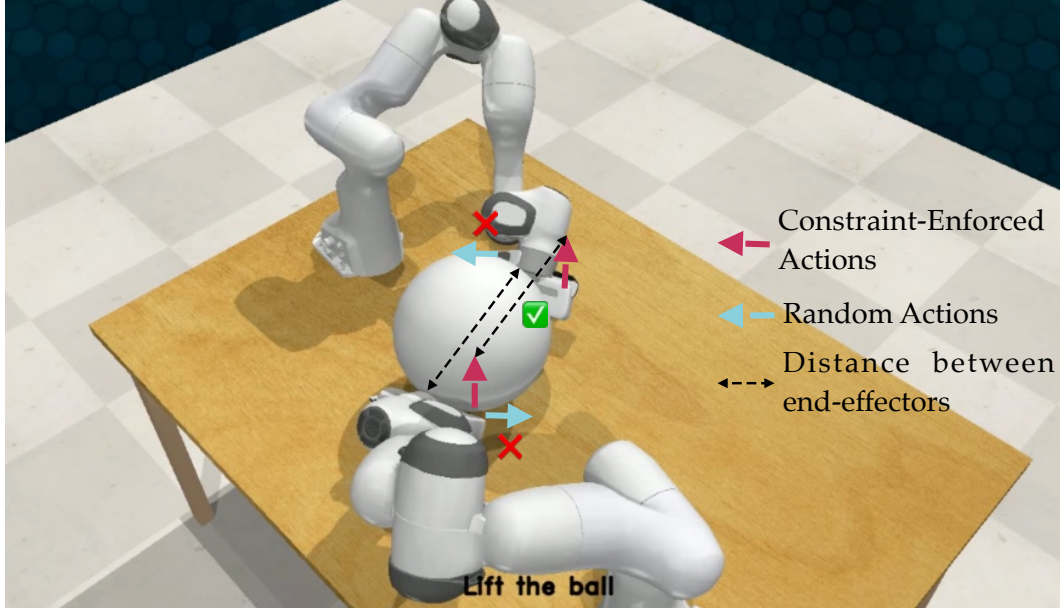


Figure 10: Visualization comparing constraint-enforced actions and random actions.

623 Figure 10 shows a visualization comparing constraint-enforced actions and random actions in the  
 624 Coordinated Lift Ball task. At this timestep, the robot arms have reached the bottom of the  
 625 ball and are about to lift it. If the next sampled actions are random (blue arrows), they may cause  
 626 the ball to fall due to an increased distance between the end-effectors (black arrows). In contrast, if  
 627 constraint-enforced actions are sampled (maroon arrows), the distance and orientations of the end-  
 628 effectors are maintained, preserving the conditions necessary for the ball to remain stable atop the  
 629 grippers. Thus, constraint-enforced actions are critical for achieving coordinated bimanual manipu-  
 630 lation.



## 631 H Examples of Synthesized Images

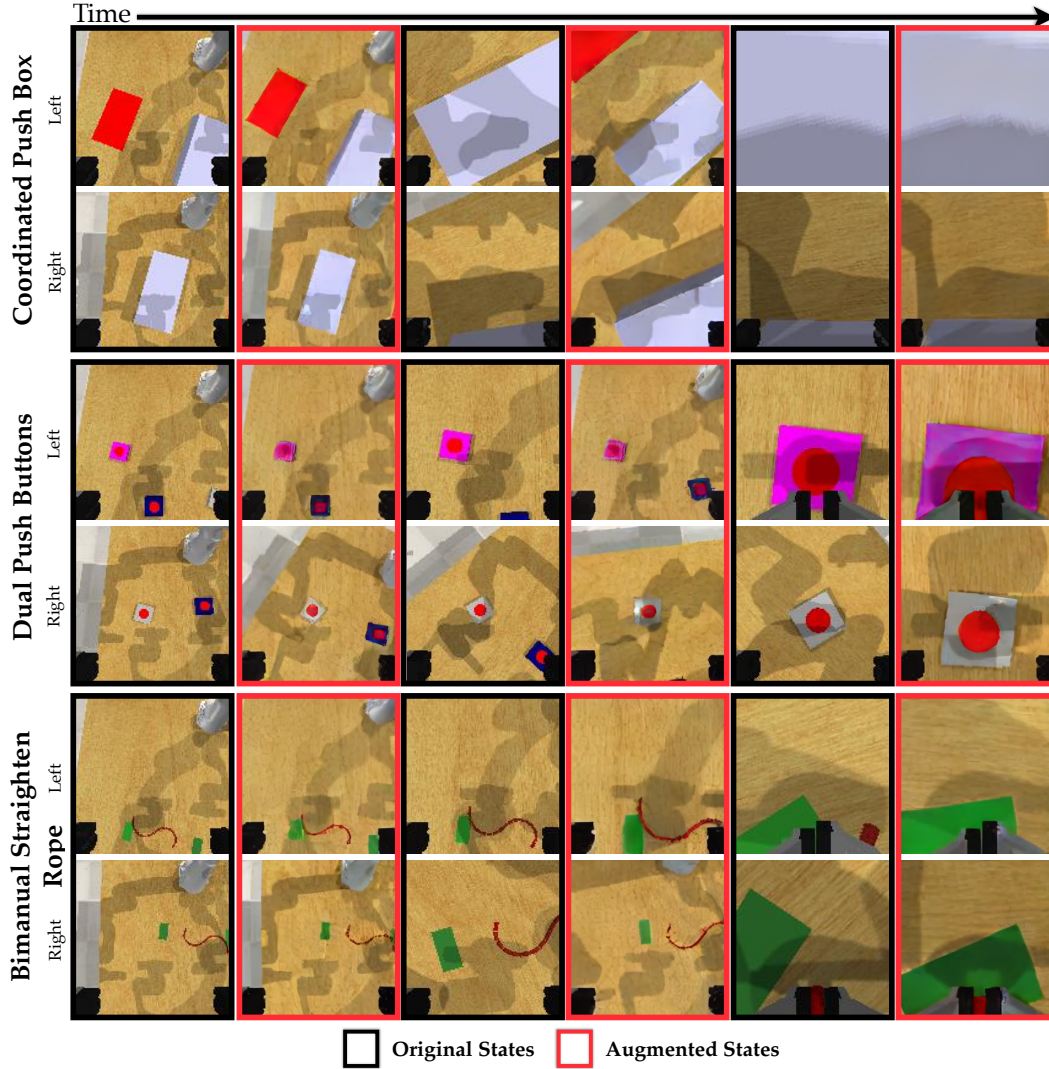


Figure 11: Examples of the original and synthesized wrist-camera images from both arms using D-CODA on Coordinated Push Box, Dual Push Buttons, and Bimanual Straighten Rope tasks in simulation. The first black column of images are the original states where the following column of red images are the augmented (perturbed original) states. All original and augmented state pairs are at the same timestep and each task is from the same episode.

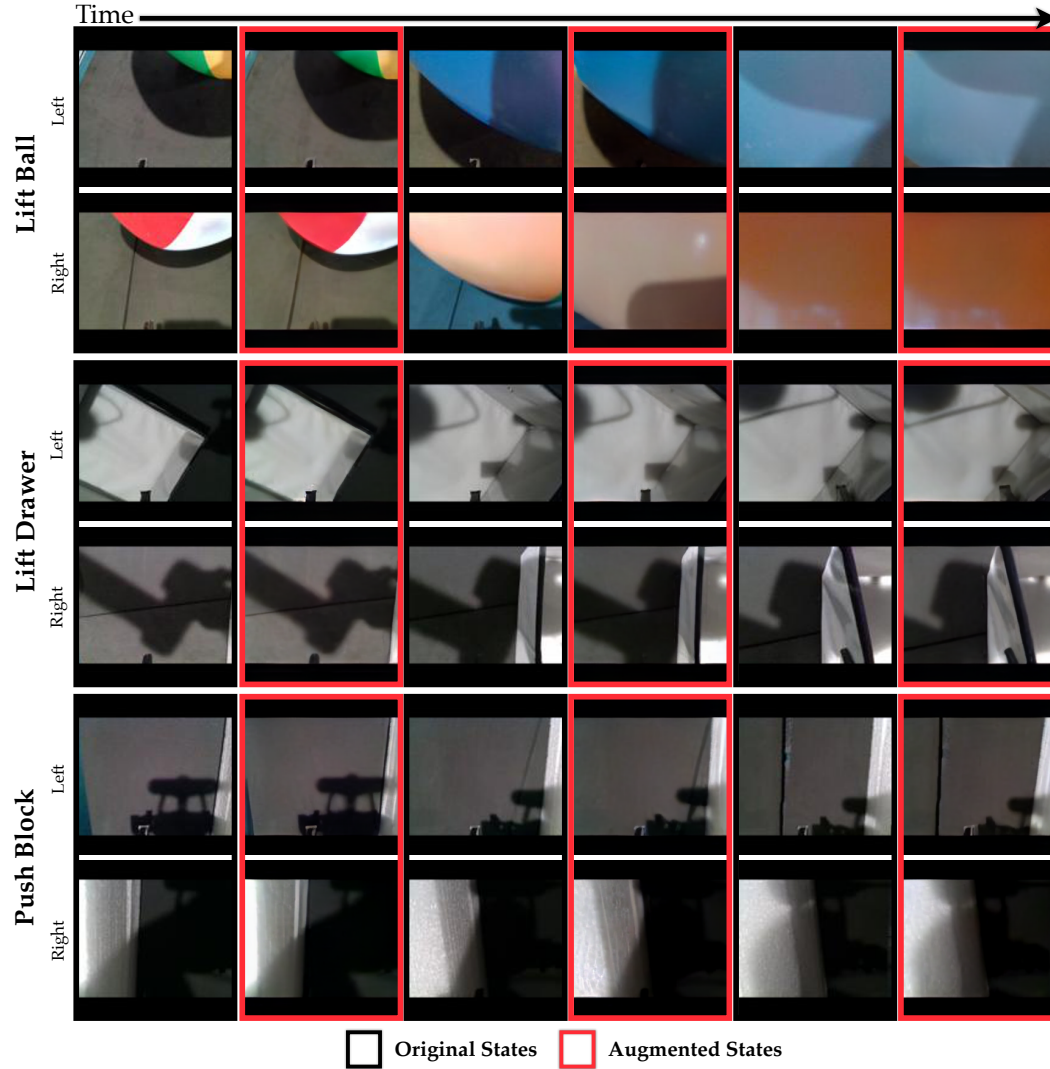


Figure 12: Examples of the original and synthesized wrist-camera images from both arms using D-CODA on the real-world **Lift Ball**, **Lift Drawer**, and **Push Block** tasks. The first black column of images are the original states where the following column of red images are the augmented (perturbed original) states. All original and augmented state pairs are at the same timestep and each task is from the same episode.