

Figure 7. **Real-World Assets and Experimental Settings.** We provide visualizations of the assets used and the hardware settings for Dual-Arm Franka Platform tasks.

## A. Additional Dataset Details

This section provides further details on our pretraining and real-world datasets used in the multi-stage training process. Furthermore, we elaborated in detail on the digital-twin data generation pipeline.

### A.1. Pretraining Data Curation

As outlined in Stage 1 of our training strategy, the action expert is pretrained on a large-scale, cross-embodiment assembly dataset. To build this dataset, we carefully aggregate and curate demonstrations from public robotic manipulation datasets, including Open X-Embodiment [61], Droid [35], and Robomind [86]. These datasets contain millions of trajectories from a diverse set of robotic platforms, sensor configurations, and task settings. However, only a subset of these demonstrations is directly relevant to the domain of robotic assembly and rearrangement. Therefore, a careful multi-stage filtering and unification process is required before pretraining. We first apply a task-level filtering pipeline designed to extract trajectories involving low-level manipulation primitives that commonly appear in assembly and rearrangement settings, such as precise object-manipulation, and general pick-place. After filtering and integration, the assembly pretraining dataset contains more than 400,000 high-quality trajectory samples, each representing a structured demonstration of object manipulation. Although this is a fraction of the raw data available in the source corpora, the curated subset is specifically optimized for assembly-centric skills while remaining a robust foundation for the action expert to learn a wide range of manipulation primitives before fine-tuning on downstream tasks.

### A.2. Real-World Data Collection

For real-world experiments, we evaluate three downstream tasks (2D LEGO Assembly, 3D LEGO Assembly, Object

Table 4. **The dataset name and sampling weight used in our mixed large-scale pretraining dataset.**

Training Dataset Mixture	
Fractal [8]	6.8%
Kuka [32]	10.5%
Bridge[19, 79]	4.9%
Taco Play [57, 69]	2.5%
Jaco Play [17]	0.4%
Berkeley Cable Routing [53]	0.2%
Roboturk [56]	2.0%
Viola [101]	0.8%
Berkeley Autolab UR5 [13]	1.0%
Toto [97]	1.7%
Language Table [55]	3.7%
Stanford Hydra Dataset [3]	3.8%
Austin Buds Dataset [100]	1.8%
NYU Franka Play Dataset [16]	0.7%
Furniture Bench Dataset [25]	2.1%
UCSD Kitchen Dataset [88]	<0.1%
Austin Sailor Dataset [59]	1.9%
Austin Sirius Dataset [46]	1.5%
DLR EDAN Shared Control [66]	<0.1%
IAMLab CMU Pickup Insert [70]	0.7%
UTAustin Mutex [72]	1.9%
Berkeley Fanuc Manipulation [99]	0.6%
CMU Stretch [58]	0.1%
BC-Z [30]	6.3%
FMB Dataset [54]	6.0%
DobbE [71]	1.2%
DROID [35]	14.2%
Stanford Kuka Dataset [38]	0.3%
Stanford Robocook Dataset [74]	0.2%
Columbia Cairlab Pusht Real [15]	<0.1%
UCSD Pick and Place	0.8%
Maniskill [24]	7.5%
Berkeley RPT [68]	<0.1%
QUT Dexterous Manipulation [11]	<0.1%
RoboSet [37]	5.2%
BridgeData V2 [79]	9.3%
RoboMind [86]	1.2%

Rearrangement) on the dual-arm Franka platform. Below, we detail the hardware configurations, task settings, and data protocols.

**Hardware Configurations.** We equip the dual-arm Franka experimental environment with two Franka Research 3 arms each with a 3D-printed UMI gripper, the configurations of which is summarized in Table 5. As shown in Figure 7, we utilize an Intel RealSense 455 camera to capture a static third-person view at the speed of 30Hz.

**Task Settings.** We provide a detailed explanation of the

Table 5. The hardware setups of the Franka Research 3, including joint position limits and velocity limit.

Joint Number	Position Limits	Velocity Limits
J1	$-166^\circ \sim +166^\circ$	$150^\circ/s$
J2	$-105^\circ \sim +105^\circ$	$150^\circ/s$
J3	$-166^\circ \sim +166^\circ$	$150^\circ/s$
J4	$-176^\circ \sim -7^\circ$	$150^\circ/s$
J5	$-165^\circ \sim +165^\circ$	$301^\circ/s$
J6	$+25^\circ \sim +265^\circ$	$301^\circ/s$
J7	$-175^\circ \sim +175^\circ$	$301^\circ/s$

assembly and rearrangement tasks and their success conditions. **2D Assembly:** The final state of the task is that LEGO blocks of different colors are randomly inserted into the same layer position on the planar board, without any stacking of blocks. Given the final 2D assembled structure as the goal, the robot arms need to pick up the LEGO blocks from both sides of the board in turn and insert them into the correct corresponding position in the center of the board. We only consider it a success if the current Lego blocks on the board match the position of the final structure at each key intermediate step. For a complete evaluation, only if all key intermediate steps are successful will it be counted as successful. **3D Assembly:** Based on the 2D Assembly Task, the 3D Assembly Task allows for more complex placement situations such as stacking between LEGO blocks. For evaluation, we do not require the placement order of LEGO blocks, but we still require that the LEGO blocks placed at each key intermediate step conform to the placement position of the corresponding color LEGO block in the final state, and the final 3D LEGO shape needs to match the given shape. **Object Rearrangement:** Unlike Assembly tasks, we use bowls, bananas, beverage cans and other common objects in life. The goal of the task is to pack the objects on the table into the box in turn and conform to the given placement pattern. In the rearrangement task, the order of placement is very important since there should be no situation where the objects in the bowl are placed before the bowl during execution. We follow the same evaluation settings as Assembly tasks at key intermediate steps and final states. For each task, the execution order is determined by the sequence in which the planning expert generates the step-by-step manual.

**Data Protocols.** For each task, we collect 100 demonstrations using 3DConnexion Spacemouse to teleoperate each Franka arm with target positions randomized on the table and box to promote data diversity. Language instructions are manually created and diversified via augmentation. Each trajectory is recorded at a frequency of 15hz, and each step contains a third-angle image shaped 640x480, the dual-arm end effector poses and the grippers discrete opening and closing. For each trajectory, we automatically filter key intermediate steps by grippers status to obtain the subgoal image,

and use pixel matching to obtain the low-level ( $U, V$ ) coordinates corresponding to each image. For all methods, we apply keyframe extraction similar to the simulation settings to construct training data [76], filtering out unnecessary jitter that could hinder precise action.

### A.3. Digital-Twin Data Generation

To train the planning expert (Stage 2) without the prohibitive cost of large-scale human annotation, we developed a high-fidelity digital-twin toolkit based on 3D Gaussian Splatting (3DGS). The pipeline consists of two main steps:

1. **Asset Reconstruction:** We first reconstruct high-fidelity 3D assets of all relevant objects, including the LEGO board, individual bricks of various colors, and the objects used in the rearrangement task. This is achieved by capturing multi-view images of each object and using them to train a 3DGS [34] model. The resulting representations are then decomposed and aligned to a unified Cartesian coordinate system for consistent spatial referencing.
2. **Iterative Scene Generation:** Given an initial state and a set of available objects, the toolkit iteratively and automatically generates intermediate task states. For LEGO assembly, it sequentially places each brick by randomly sampling a valid position on the board. At each intermediate step, we render a photorealistic image of the current scene from a fixed front-view camera perspective. This process provides the necessary data for training the planning expert: the rendered image serves as the subgoal image, the brick’s board position provides the ( $U, V$ ) coordinates, and a corresponding textual description (e.g., “Yellow blocks assemble”) is generated via templates.

This automated pipeline enabled us to generate a dataset of over 10,000 frames for each task, providing the rich data needed to effectively pretrain the planning expert. More visualizations of the generated sim-to-real data are shown in Figure 8 and Figure 9.

## B. Additional Experimental Details

In this section, we present quantitative results of simulation experiments on RL Bench, which provide solid and strong evidence of our model’s ability to manipulate on general tasks, retaining the basic capabilities of the VLA model. Furthermore, we report the additional ablation studies on the impact of manual generation quality and token sequence arrangement on the model’s action accuracy. Finally, we conduct additional real-world experiments with extended rollout evaluations and more comprehensive baseline comparisons.

### B.1. Simulation Experiment

We evaluate ManualVLA on the RL Bench [29] benchmark, comparing it against state-of-the-art (SOTA) VLA baselines. The results demonstrate ManualVLA’s robust capabilities in predicting future images and generating precise actions.

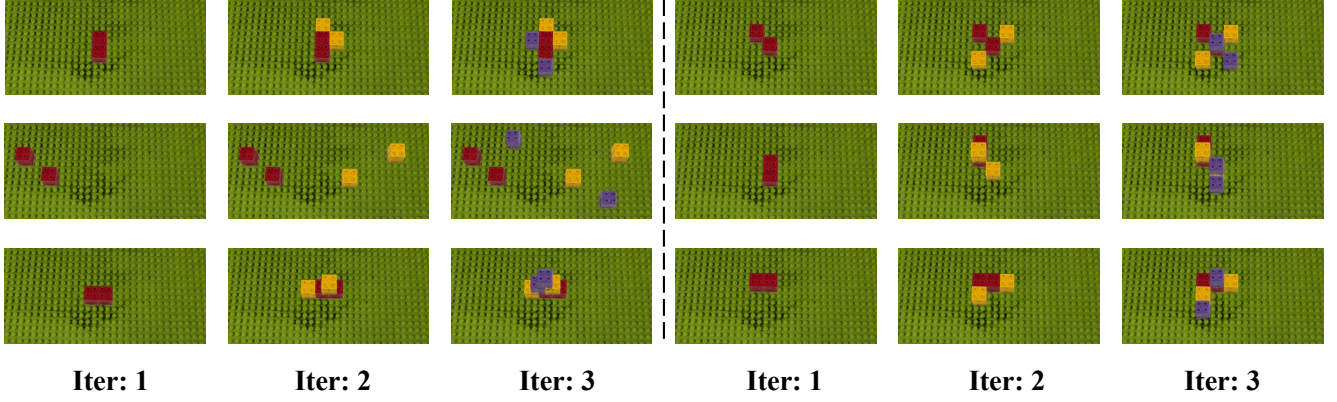


Figure 8. **Iterative manual generation examples for LEGO Assembly.** Each row shows a sequence where two bricks are progressively stacked per iteration. Scenes are rendered at each step using our digital-twin toolkit.

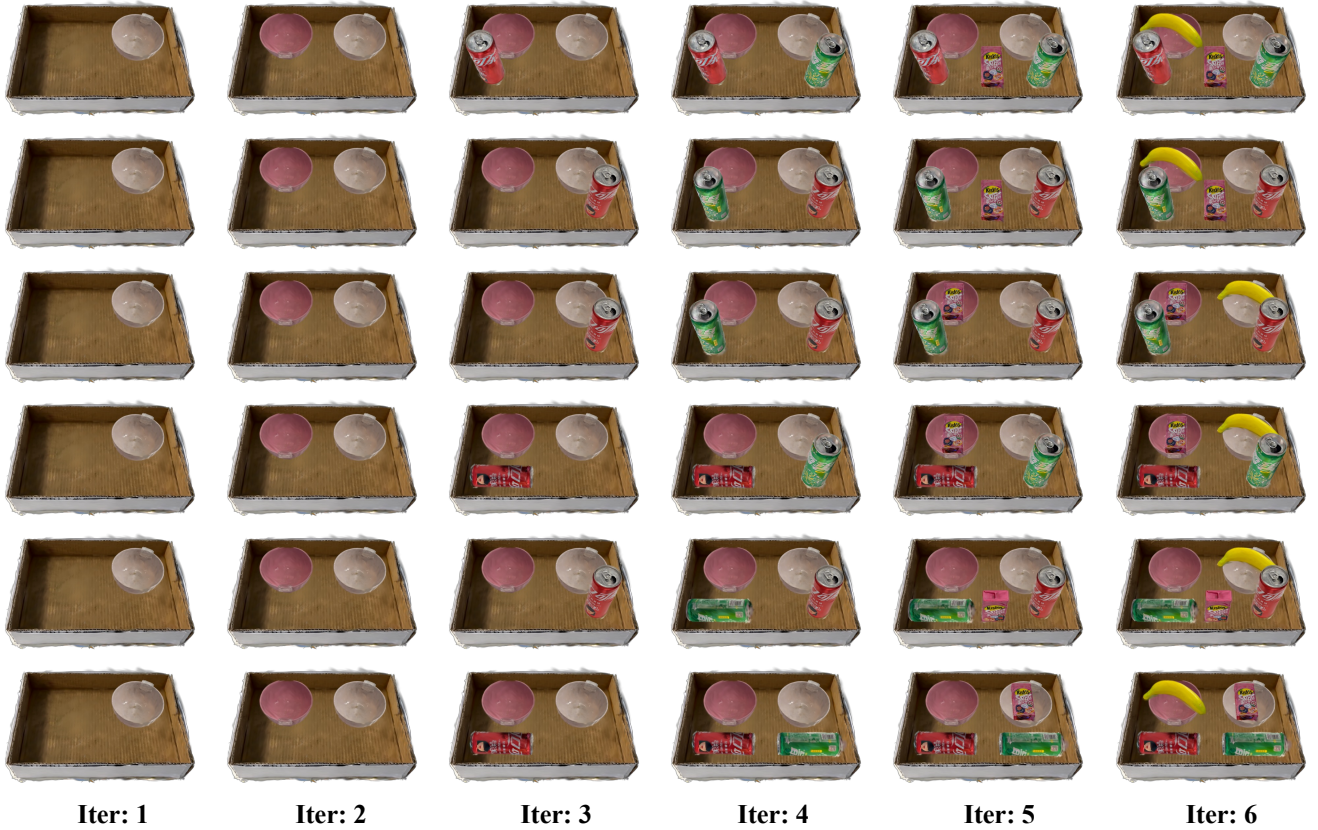


Figure 9. **Iterative manual generation examples for Object Rearrangement.** Each row shows a sequential process where objects are placed into the box one at a time. Scenes are rendered at each step using our digital-twin toolkit.

### B.1.1. Simulation Benchmark

To assess the fundamental manipulation capabilities of our method across common manipulation tasks, we conduct experiments on 10 tasks in the RLBench [29] benchmark based on the CoppeliaSim simulator. The task suite includes

*Close box, Close Laptop, Toilet seat down, Sweep to dustpan, Close fridge, Phone on base, Take umbrella out, Take frame off hanger, Place wine at rack, and Water plants.* All tasks are executed on a Franka Panda robot equipped with a front-view RGB camera to get the visual input. We collect the data

Table 6. **Comparison of ManualVLA and baselines on RL Bench.** We train all methods in the multi-task setting [76] and report the success rates (S.R.) and variances (Var.).

Models	Close box	Close laptop lid	Toilet seat down	Sweep to dustpan	Close fridge	Phone on base	Umbrella out	Frame off hanger	Wine at rack	Water plants	Mean S.R. & Var
FAST [63]	0.85	0.70	<b>0.90</b>	0.45	0.60	0.15	0.15	0.25	0.45	0.15	0.47 $\pm$ 0.03
$\pi_0$ [7]	0.85	<b>0.95</b>	<b>0.90</b>	0.85	0.80	0.25	0.20	0.65	0.65	0.25	0.63 $\pm$ 0.01
$\pi_{0.5}$ [27]	<b>0.90</b>	0.80	<b>0.90</b>	0.50	0.60	0.15	0.25	0.35	<b>0.75</b>	<b>0.35</b>	0.56 $\pm$ 0.03
CoT-VLA [96]	<b>0.90</b>	0.85	<b>0.90</b>	0.60	0.70	0.20	0.30	0.55	0.55	0.30	0.59 $\pm$ 0.03
<b>ManualVLA (ours)</b>	<b>0.90</b>	0.90	<b>0.90</b>	<b>1.00</b>	<b>0.85</b>	<b>0.30</b>	<b>0.50</b>	<b>0.70</b>	0.65	<b>0.35</b>	<b>0.70</b> $\pm$ 0.02

by following pre-defined waypoints and utilizing the Open Motion Planning Library [77]. Building upon the frame-sampling technique employed in previous studies [23, 31, 76], we construct a training dataset where each task contains 100 trajectories. To generate ground-truth ( $U, V$ ) labels, we follow the key-frame extraction procedure from prior work. Specifically, we extract the end-effector poses of the key frames and then use the camera parameters to project their world-coordinate positions into ( $U, V$ ) coordinates.

### B.1.2. Training and Evaluation Details

We compare ManualVLA against four state-of-the-art (SOTA) VLA models, including FAST [63],  $\pi_0$  [7],  $\pi_{0.5}$  [27], and CoT-VLA [96]. While the former three adopt robust action-generation paradigms, CoT-VLA conditions on the final goal image and additionally predicts future subgoal images. Specifically, FAST [63] utilizes autoregressive action outputs,  $\pi_0$  [7] and  $\pi_{0.5}$  [27] employ flow matching, and CoT-VLA [96] combines autoregressive image generation with diffusion-based action prediction. For all baselines, we initialize with the official pretrained parameters and strictly adhere to their original fine-tuning configurations. To ensure a fair comparison, we align the subgoal supervision in CoT-VLA to match the formulation used in ManualVLA. For ManualVLA’s input, the single-view RGB image is resized to  $384 \times 384$ , with text instructions derived directly from the simulation environment and the robot state is aligned with the predicted actions. ManualVLA model is trained for 500 epochs using the AdamW optimizer [51] and CosineAnnealingLR [52] on 8 NVIDIA H20 GPUs, with mixed-precision training employed. Following [23, 40], we evaluate all methods using 20 rollouts from the latest epoch checkpoint, repeating the evaluation three times for each task and reporting the average success rate along with the variance.

### B.1.3. Quantitative Results

As presented in Table 6, ManualVLA achieves an average success rate of 70% across 10 diverse tasks, surpassing the previous SOTA methods  $\pi_0$  [6] and CoT-VLA [96] by margins of 7% and 11%, respectively. Specifically, ManualVLA attains superior performance on 8 out of 10 tasks, highlighting the advantage of ManualCoT strategy in guiding precise action generation. By generating sub-goal images and

constructing visual prompt images, ManualVLA effectively leverages the fine-grained affordance guidance provided by explicit CoT reasoning. Furthermore, the MoT architecture, equipped with a shared attention module, enables robust task understanding and action generation conditioned on the subgoal manual within the latent space. Through the integration of both explicit and implicit CoT reasoning, ManualVLA demonstrates substantial improvements in tasks requiring precise actions, such as *sweep to dustpan* and *take out umbrella*, compared to  $\pi_0$  and  $\pi_{0.5}$ .

## B.2. Additional Ablation Studies

This section will present additional ablation studies to further validate our design choices.

Table 7. **Comparison of manual generation quality impact on action generation.**

Training Frames	PSNR $\uparrow$	2D LEGO Assembly			
		2 bricks $\rightarrow$ 2 bricks $\rightarrow$ 2 bricks $\rightarrow$ S.R.			
0.5K	25.71	0.35	0.25	0.20	0.20
1K	26.61	0.45	0.35	0.30	0.25
3K	27.16	0.65	0.65	0.60	0.60
6K	28.29	0.85	0.80	0.80	0.80
10K	29.01	<b>0.95</b>	<b>0.90</b>	<b>0.85</b>	<b>0.85</b>

### B.2.1. Impact of Manual Generation Quality on Action

To evaluate the robustness of the action expert under varying manual-generation quality, we compare five versions of our planning expert trained on datasets of 0.5K, 1K, 3K, 6K and 10K frames, respectively, for the 2D LEGO Assembly task. All these training frames are generated using our high-fidelity digital-twin toolkit. These planning experts produce manuals with differing levels of fidelity, quantified by the PSNR of the generated subgoal images. We condition ManualVLA on these manuals and measure the resulting task success rate over 20 rollouts, as reported in Table 7. The results indicate that low-quality subgoal images lead to substantial error accumulation during action generation, significantly degrading overall performance. Once the training set reaches 1,000 samples, yielding manuals with PSNR above 27, the action expert exhibits stable and reliable behavior. This trend high-



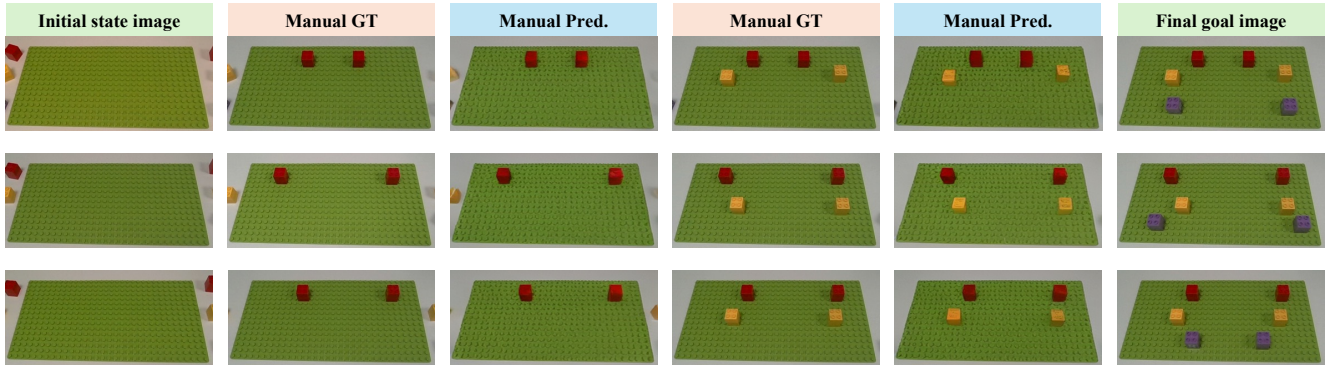


Figure 10. **Iterative manual generation examples for 2D LEGO Assembly.** Pred refers to the predictions generated by our model, while GT denotes the ground truth in the test set.

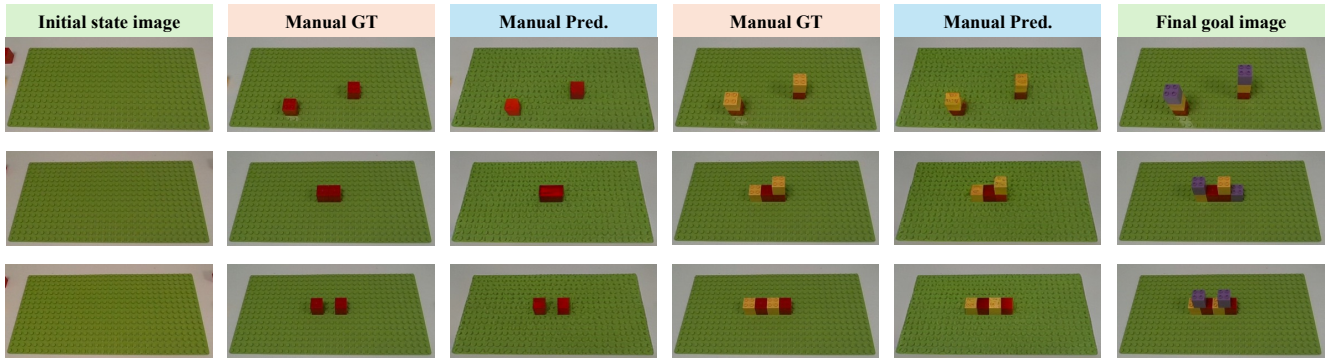


Figure 11. **Iterative manual generation examples for 3D LEGO Assembly.** Pred refers to the predictions generated by our model, while GT denotes the ground truth in the test set.

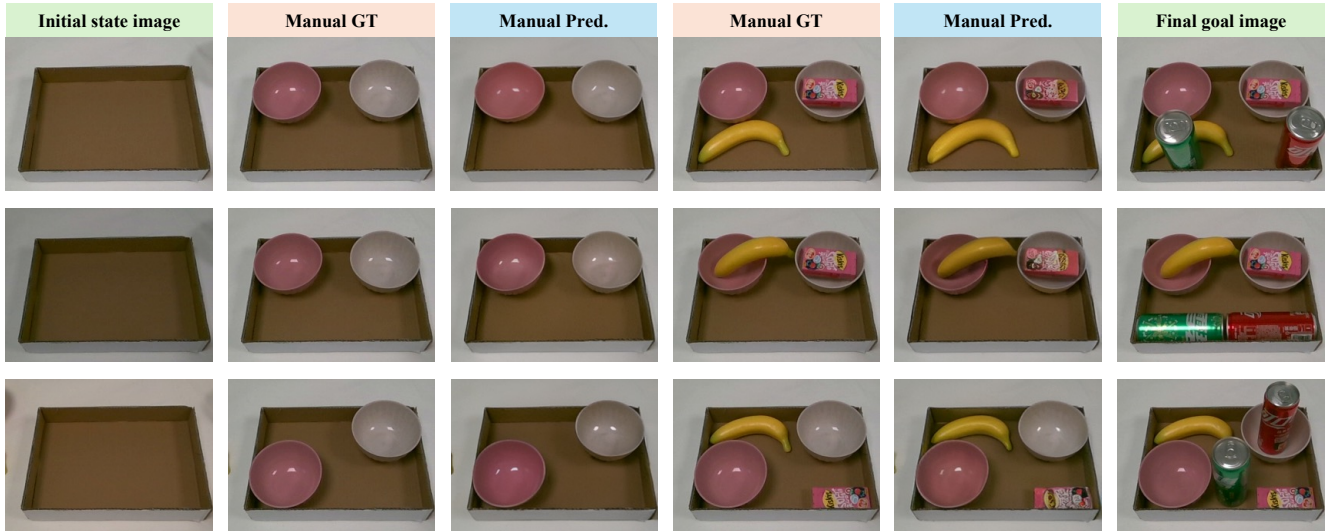


Figure 12. **Iterative manual generation examples for objects rearrangement.** Pred refers to the predictions generated by our model, while GT denotes the ground truth in the test set.

lights ManualVLA’s robustness: when the planning expert is sufficiently trained, its explicit and implicit chain-of-thought reasoning reliably supports consistent action generation.

### B.2.2. Impact of Token Sequence Arrangement

To further analyze how the ordering of multimodal tokens in the generated manual influences action generation, we evaluate four different sequence arrangements, as shown in Table 8. The modalities contained in the manual serve complementary purposes: the text instruction provides high-level semantic goals, the coordinate tokens ( $U, V$ ) specify the future spatial locations of the objects to be manipulated, and the generated manual image offers step-wise visual cues synthesized by the planning expert. Finally, the visual prompt image is conditioned on the generated coordinates ( $U, V$ ), and therefore we place it at the end of the token sequence. This ordering naturally forms a pipeline in which the model first performs implicit CoT reasoning, followed by explicit CoT reasoning, before producing the final action sequence. Because the effectiveness of the action expert depends on how well it integrates semantic and visual information, the ordering of these tokens can substantially affect downstream policy performance. Our study on different generation orders of the three types of manual information reveals that the sequence of generating text first, then coordinates, and finally subgoal images yields the best task success rate. Meanwhile, the other sequence configurations introduce only minor performance degradation, demonstrating both the robustness of ManualVLA and the critical role of the combined implicit–explicit CoT reasoning process in enabling strong action-generation performance.

Table 8. **Comparison of different token sequence order impact on action generation.** Here,  $T$  and  $P$  denote the textual description and coordinate pairs ( $U, V$ ) in the manual, while  $I_{\text{subgoal}}$  and  $I_{\text{prompt}}$  refer to the subgoal image and the visual prompt image.

Token Sequence	2D LEGO Assembly			
	2 bricks	→ 2 bricks	→ 2 bricks	→ S.R.
$P \rightarrow I_{\text{subgoal}} \rightarrow T \rightarrow I_{\text{prompt}}$	0.90	0.85	0.80	0.80
$I_{\text{subgoal}} \rightarrow P \rightarrow T \rightarrow I_{\text{prompt}}$	0.75	0.75	0.70	0.70
$T \rightarrow I_{\text{subgoal}} \rightarrow P \rightarrow I_{\text{prompt}}$	0.85	0.80	0.80	0.80
$T \rightarrow P \rightarrow I_{\text{subgoal}} \rightarrow I_{\text{prompt}}$	<b>0.95</b>	<b>0.90</b>	<b>0.85</b>	<b>0.85</b>

## B.3. Additional Real-World Experiments

### B.3.1. Additional Evaluations

To rigorously evaluate the stability of our approach, we conducted two additional sets of 20 rollouts on unseen goal states, bringing the total to 60 evaluation rollouts per task. As shown in Tab. 9, ManualVLA consistently demonstrates strong performance, achieving mean success rates of 83.3% for 2D LEGO Assembly, 61.6% for 3D LEGO Assembly, and 68.3% for Object Rearrangement. The low average standard deviation of 2.36% across these trials further underscores the reliability of our method when confronted with

Table 9. **Additional Evaluation.** We report the mean success rate and average standard deviation for each task over 60 rollouts.

Task	2D LEGO	3D LEGO	Object
ManualVLA	83.3%±2.36%	61.6%±2.36%	68.3%±2.36%

Table 10. **Experiments on Text-Goal Conditioning.** We report the success rates on the newly introduced Simple LEGO task, comparing ManualVLA against baselines when the objective is specified entirely via text.

Method	ManualVLA	VLM+ $\pi_{0.5}$	$\pi_{0.5}$
Goal Text	70%	50%	25%

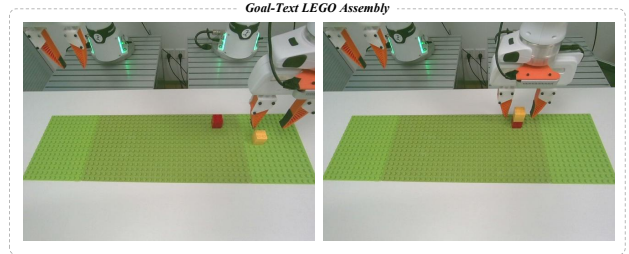


Figure 13. **Text-Goal Conditioned LEGO Assembly Task.** Visualization of the text-conditioned setup, where the robot is instructed to execute semantically unambiguous placement configurations.

diverse final configurations.

### B.3.2. Additional Baseline Comparisons

We expand our comparative analysis on the 2D LEGO task by evaluating ManualVLA (which achieves an 85% success rate) against two additional baselines: Hi-Robot [75], a hierarchical subgoal-text method, and RT Goal-Image (RTGI) from RT-Sketch [78], a goal-image-conditioned approach. Because RTGI lacks the capability to generate subgoal images, we report its performance under two conditions: utilizing only the final goal image (achieving a 40% success rate) and utilizing both the final and subgoal images (achieving a 50% success rate). Hi-Robot similarly yields a lower score, primarily because complex final LEGO configurations are inherently difficult to specify precisely through language alone. These comparisons highlight that our ManualCoT reasoning process, empowered by the cross-task shared attention design, couples long-horizon planning and fine-grained action much more effectively than existing goal-conditioned baselines. Furthermore, we investigate the distinction between goal-text and goal-image conditioning on a newly introduced Goal-Text LEGO Assembly task, where task objectives are easily articulated in language (e.g., “insert the yellow block on top of or to the left of the red block,” as visualized in Fig. 13). As detailed in Tab. 10, when task goals are semantically well-defined, ManualVLA achieves a 70% success rate under goal-text conditioning. This significantly outperforms the

hierarchical VLM+ $\pi_{0.5}$  baseline (50%) and the standard  $\pi_{0.5}$  model (25%). These results indicate that both text and image modalities serve as highly effective conditioning signals for our architecture, and that ManualVLA remains highly robust and flexible when processing diverse instruction formats.

## C. Additional Qualitative Results

This section provides more qualitative results for manual generation and real-world robot rollouts.

### C.1. Manual Generation Visualization

Figure 10, Figure 11, and Figure 12 provide additional visualizations of the manuals generated by our planning expert across all three downstream tasks. These examples showcase the model’s ability to generate structured and interpretable intermediate states that accurately guide the subsequent action generation. For the LEGO assembly tasks, ManualVLA sequentially reconstructs the correct brick placements and colors, demonstrating precise step-wise reasoning. Similarly, for object rearrangement, it progressively generates subgoals that accurately capture the spatial relationships between objects, moving step-by-step toward the final goal configuration. Overall, these results highlight ManualVLA’s strong intermediate reasoning capabilities, establishing a reliable foundation for the action expert to generate accurate actions.

### C.2. Real-World Rollout Visualization

The qualitative rollouts in Figure 14 further corroborate our quantitative findings, illustrating keyframes of the dual-arm real-world execution processes. The visualizations demonstrate that ManualVLA can follow the internally generated manuals to reliably guide the action expert in producing precise grasping, insertion, and placement motions. In both the 2D and 3D LEGO assembly tasks, compared with the final goal image, the robot consistently maintains accurate brick placement throughout all stages. For the object-rearrangement task, also compared with the final goal image, it stably manipulates objects with varying shapes and occlusions. These results collectively validate ManualVLA’s strong action generation capabilities, demonstrating its potential as a robust policy for real-world, long-horizon robotic manipulation.

## D. Failure Case Analysis

Although ManualVLA demonstrates strong overall performance, it is not without limitations. Through our experiments, we identified two primary failure modes, as illustrated in Figure 15:

1. **Occasional Erroneous LEGO Placement.** While ManualVLA is generally successful in accomplishing the

LEGO assembly task, it can still produce incorrect placements. As shown in Cases 1 and 2 of Figure 15, the system places the yellow bricks in incorrect positions due to model errors. Notably, however, the system is often able to recover from such mistakes and correctly place subsequent bricks.

2. **Placement Errors Under Large Rotation Angles.** In the Objects Rearrangement task, certain scenarios require the robot arm to perform large rotations to achieve the correct placement orientation. ManualVLA may fail in these situations, as illustrated by Case 3 in Figure 15, where the robot arm fails to place the spirit can into the box. We hypothesize that these failures stem from the limited number of such extreme rotation cases in the training data.

## E. Supplementary Video

A supplementary video is included with the supplementary material. It presents real-world robot rollouts for all three tasks, visualizations of the generated subgoal manuals, and an overview of our proposed method. Notably, for the complex 3D assembling task, we recollected teleoperation data and retrained the action expert to enable alternating left–right arm control, thereby preventing collisions during inference.





Figure 14. **Real-World Task Execution Progress Visualization.** We provide visualizations of three real world tasks including assembly and rearrangement evaluated on dual-arm Franka robot platform.

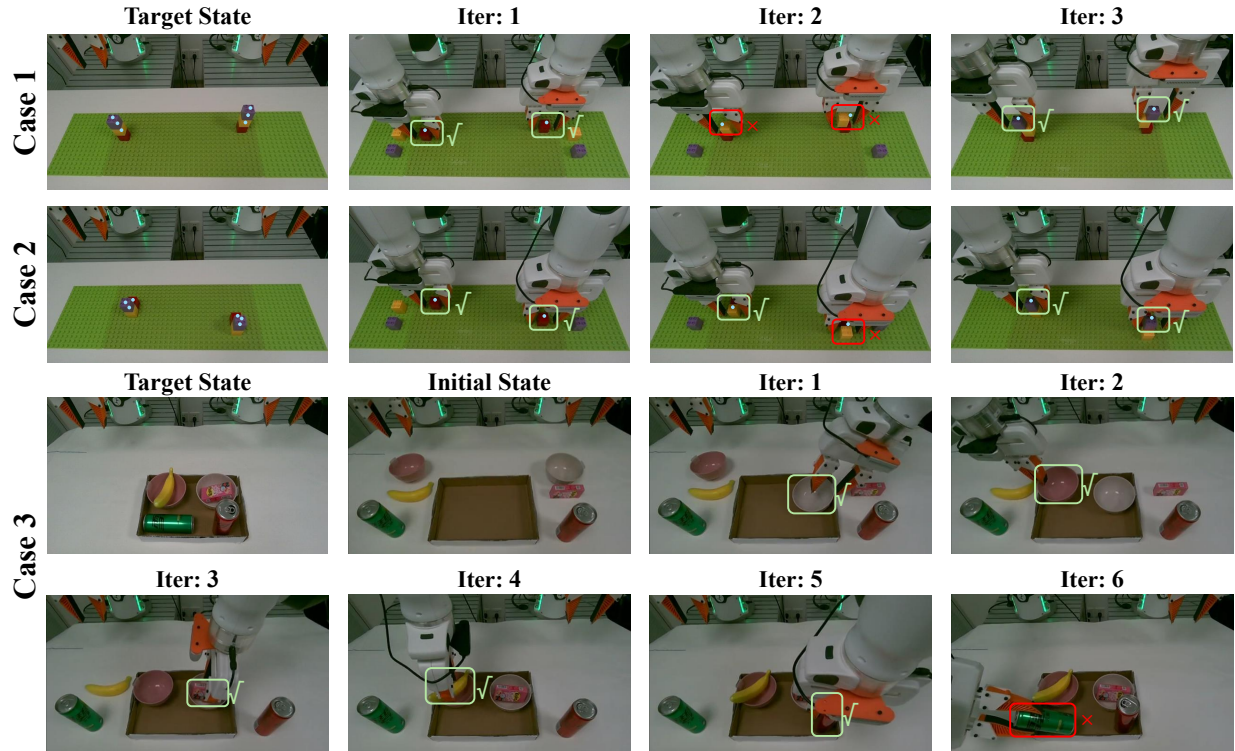


Figure 15. **Failure cases in our two tasks: LEGO assembly and objects rearrangement.** The top two rows illustrates two LEGO failure cases and the bottom two rows shows a failure case of objects rearrangement task.