Spatial-Temporal Graph Diffusion Policy with Kinematic Modeling for Bimanual Robotic Manipulation

Supplementary Material

We provide a more comprehensive **tasks descriptions** in Section A, encompassing both simulated environments and real-world scenarios. In Section B, we elaborate on the **implementation details**, including code base of baseline methods, and hyperparameter configurations for the backbone, graph encoder modules and optimization process. Notably, to thoroughly validate efficacy of KStar Diffuser, we conduct **extensive ablation studies** analyzing the impact of demonstration quantity, action chunking size, observation history length, and the control learning objective coefficient λ. The result is presented in Section C. Finally, we present more **qualitative analysis** in Section D.

A. Task Descriptions

We conduct extensive experiments on both simulated tasks and real-world tasks to evaluate the effectiveness of our proposed KStar Diffuser. Specifically, we selected five tasks from RLBench2, ranging from basic symmetrical tasks to advanced coordination-requiring tasks, including `push_box`, `lift_ball`, `handover_item_easy`, `sweep_dustpan`, and `pick_laptop`. For real-world evaluations, we created a similar setup with two bimanual tasks: `lift_plate` and `handover_item_easy`. We present the details about both simulated and real-world tasks in Table A.

Table A. Tasks Details.

Task	Duration	# Keyframes	Instruction
Simulated Tasks			
`push_box`	4.33s	2.1	*"Push the box to the red area."*
`lift_ball`	4.40s	4.0	*"Lift the ball."*
`handover_item_easy`	7.17s	7.5	*"Handover the item."*
`sweep_dustpan`	4.93s	7.3	*"Sweep the dust to the pan."*
`pick_laptop`	3.97s	7.2	*"Pick up the notebook."*
Real-world Tasks			
`lift_plate`	6.37s	3.4	*"Lift the plate."*
`handover_item_easy`	9.52s	8.6	*"Handover the item."*

A.1. Simulated Tasks

`push_box`. As illustrated in Figure A(a), the task requires the robot to utilize both arms to push a heavy box, weighing 50 kg, and transport it to a designated target area through a fix trajectory. The completion of the task is defined as successfully moving the box to the specified location. The scenario involves two key elements: a large box and a target area, with the primary challenge being the considerable weight of the box, which exceeds the capacity of a single arm to manage effectively. Notably, this task necessitates the use of both arms simultaneously, as a single robot is incapable of accomplishing it independently.

`lift_ball`. As depicted in Figure A(b), the task entails the robot using both arms to lift a large ball, achieving a minimum height of 0.95 meters to meet the success criteria. The object in this task is the large ball, presenting a significant coordination challenge. Due to the ball's size and the inability of the gripper to securely grasp it, the operation relies on coordinated non-prehensile manipulation, demanding precise synchronization of the arms during the lifting process. This task cannot be accomplished by a single robot because of the object's dimensions.

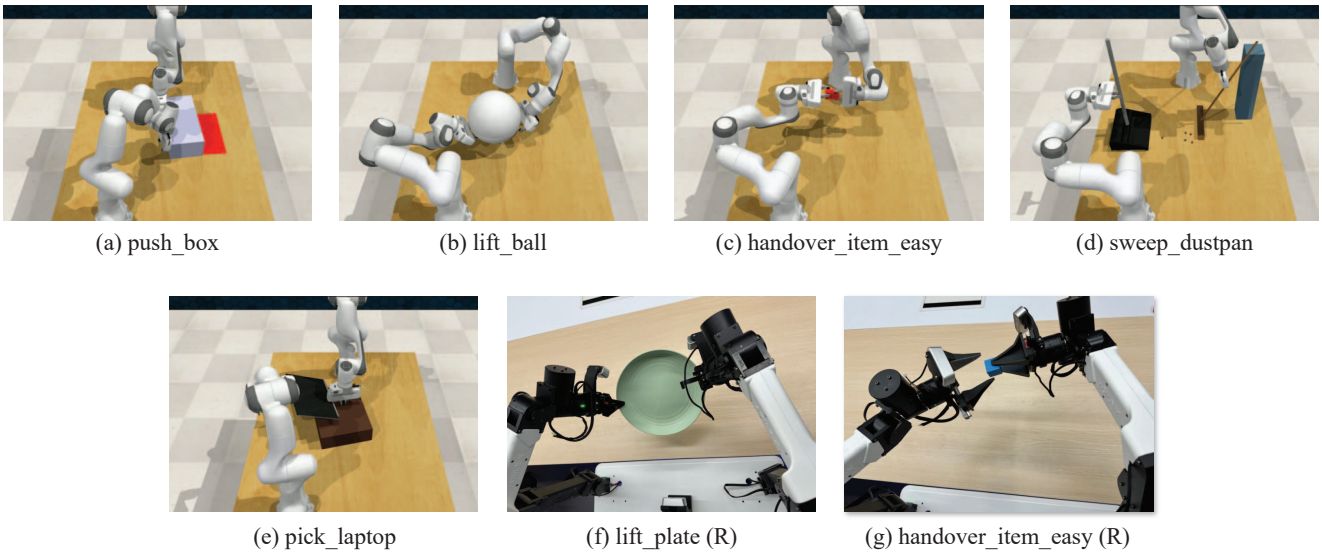
`handover_item_easy`. The task requires the robot to handover a red item by utilizing one arm to securely grasp and lift the item to a height of 80 cm while ensuring the other arm remains idle and unengaged, as shown in Figure A(c). The object involved is a single red block, and the key challenge lies in coordinating the handover process effectively. Successful completion is determined when the item is accurately identified, grasped, and positioned at the required height with no actions performed by the idle arm.

`sweep_dustpan`. It is shown in Figure A(d) that the task involves the robot using a broom to sweep dust into a dust pan, requiring precise coordination of the sweeping motion to effectively collect the dust. The objects involved include a broom, a dust pan, supporting objects, and the dust itself. Successful completion is defined as all the dust being gathered inside the dust pan. The primary challenge lies in the accuracy and control of the sweeping motion to ensure that the dust is properly directed into the pan.

`pick_laptop`. As illustrated in Figure A(e), the task requires the robot to pick up a notebook placed on top of a block by first manipulating it into a position suitable for grasping. This involves performing non-prehensile actions, such as pushing or sliding, to adjust the notebook's orientation before securely grasping and lifting it off the block. The objects involved are a notebook and a block. Successful completion is defined as the robot lifting the notebook off the block. Although the task can be performed with a single robotic arm, precise coordination is essential for effective manipulation.

A.2. Real-world Tasks

`lift_plate`. As shown in Figure A(f), the task involves the robot using both arms to lift a plate, maintaining an elevated position for over 3 seconds to meet the success criteria. The target object is a plate requiring coordinated



(a) push_box (b) lift_ball (c) handover_item_easy (d) sweep_dustpan

(e) pick_laptop (f) lift_plate (R) (g) handover_item_easy (R)

Figure A. The visualization of simulated tasks and real-world tasks. The task with "(R)" means the real-world tasks.

manipulation. Due to the plate's width and the need for stable control, the operation demands precise bimanual manipulation, requiring synchronized lifting motions from both arms. This task cannot be accomplished by a single robot arm given the plate's dimensions and stability requirements.

`handover_item_easy`. Similar to the corresponding simulated task, this task requires the robot to handover a blue item by utilizing one arm to securely grasp and lift the item as shown in Figure A(g). However, the robot do not need to lift the item to a height of 80 cm, but 30cm instead considering the security, while ensuring the other arm remains idle and unengaged, The object involved is a blue block, and the key challenge is as the same as that of in simulation. Successful completion is determined when the item is accurately identified, grasped, and positioned at the required height with no actions performed by the idle arm.

B. Implementation Details

B.1. Details of KStar Diffuser

B.1.1. Backbone

Vision Branch. Following the existing work [**?** **?**], we employ multiview RGB-D observation as visual input, where the resolution of RGB images is $3\times256\times256$ and the depth data is processed into point clouds (65536×3) using the camera's intrinsic and extrinsic parameters. The camera views contain front, overhead, right_over_shoulder, left_over_shoulder, right_wrist, left_wrist. We uniformly use a Vision Transformer which is trained from scratch as the vision encoder.

Language Branch. For the language instruction, we encode the instruction with CLIP's language encoder[1]. Specifically, the input sentence is preprocessed by the CLIP's tokenizer and then encoded to a sequence of dimensions $\mathbb{R}^{77\times512}$. We use its hidden state of "`[CLS]`" as the textual feature. It is worth noting that we extracted all text features in advance. Therefore, throughout the training phase, the weights of the text encoder do not participate in the gradient calculation of backpropagation.

Fusion Module. The fusion module begins with upsampling the visual features, followed by feature-wise modulation (FiLM) to integrate textual features, projecting them into a high-dimensional semantic space. This hierarchical fusion process is performed iteratively l_{FiLM} times, where we empirically set l_{FiLM} to 3.

The relevant hyperparameters of the backbone are presented in Table B.

Table B. The hyperparameters of Backbone.

	Vision	Text
patch size	16	`N/A`
hidden size	64	512
# layers	6	12
# heads	8	8
intermediate size	64	2048
dropout ratio	0.1	0.0
activation	`lrelu`	`quick gelu`
trainable	✓	×

[1]https://huggingface.co/openai/clip-vit-base-patch32

B.1.2. Spatial-Temporal Graph

We construct the spatial graph based on the URDF file of robotic arms. In the simulation setup, we use two Franka Panda arms, each with 7 joints, resulting in a total of 14 joints. In the real-world setup, we employ a bimanual ALOHA device, which has 12 joints, with 6 joints per arm. Therefore, the number of nodes and edges in the spatial graph is set to 14 and 12 for the simulation, respectively, and 12 and 10 for the real-world setup.

For the dynamic spatio-temporal graph, we combine the spatial graphs of three consecutive timesteps and add inter-timestep edges connecting the same joint node across different timesteps. In simulated tasks, this results in a graph with 42 nodes and 33 edges. In real-world tasks, the graph has 36 nodes and 28 edges.

We use the Graph Convolutional Graph (GCN) encoder to obtain the spatial-temporal graph representation. The detailed hyperparameters are presented in Table C.

Table C. The hyperparameters of GCN Encoder.

	Simulation	Real-world
# nodes	42	36
# edges	36	28
node dimension	19	19
hidden size	128	128
intermediate size	128	128
# layers	4	4

B.1.3. Optimization Details

During training, we follow the setup of Diffusion Policy, using the DDPM scheduler to forward and denoise, where the step of forward process and reverse process is set to 100 and 1, respectively. Table D shows the training hyperparamters.

Table D. The training hyperparameters.

	Values
batch size	64
learning rate	$2e-4$
warmup step	5 k
weight decay	$1e-6$
lr scheduler	`cosine`
training step	150 k
Optimizer	`AdamW`

B.2. Code Base

The code bases employed for our evaluations are detailed as follows:

- ACT: https://github.com/markusgrotz/peract_bimanual
- RVT-LF: https://github.com/markusgrotz/peract_bimanual
- PerAct-LF: https://github.com/markusgrotz/peract_bimanual
- PerAct2: https://github.com/markusgrotz/peract_bimanual
- DP-J: https://github.com/real-stanford/diffusion_policy
- DP3: https://github.com/YanjieZe/3D-Diffusion-Policy

C. Extensive Ablation Studies

To evaluate KStar Diffuser more comprehensively, we conduct following extensive experiments.

(1) The Effects of Demonstration Quantity. Given the critical role of demonstration quantity in imitation learning, we conduct an ablation study by training the KStar diffuser with 50 demonstrations, with results reported in Table E. Additionally, we provide a comparative analysis of policy performance across varying numbers of demonstrations (20, 50, and 100), as shown in Figure 5. The results demonstrate **a clear positive correlation between demonstration quantity and policy performance**. With 20 demonstrations, the policy achieves basic task completion capabilities. Increasing to 50 demonstrations yields significant performance improvements, with success rates rising by around 4.5% across the task suite. The upward trend continues as we scale to 100 demonstrations, indicating that the policy benefits from larger demonstration sets. These findings suggest that expanding the demonstration dataset consistently enhances the policy's ability to learn and generalize manipulation tasks.

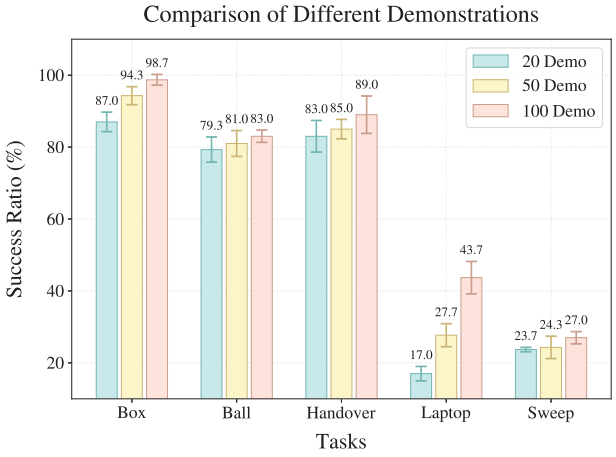


Figure B. The comparison of different number of demonstrations.

(2) The Effects of Action Chunking Size. As mentioned in previous work [**? ?**], action chunking prediction serves as an effective approach to address the multimodality challenges inherent in robotic manipulation tasks. To empirically verify this mechanism within our framework, we conducted an evaluation of the KStar Diffuser under various action chunking size configurations. Our experimental setup explores three distinct action chunking strategies: 1) single-step prediction where only the next optimal pose is gen-

Table E. The experimental result on simulated tasks. We train the policy with the setting of different training demonstrations, *i.e.* [20, 100], to test its capability comprehensively. The best results are in **bold**. Each result is reported with three seeds on average.

	Push Box	Lift Ball	Handover Item (easy)	Pick Laptop	Sweep Dustpan	Overall
Demonstration Quantity						
KStar Diffuser (*num demos=20*)	79.3 ±3.5	87.0 ±2.7	23.7 ±0.6	17.0 ±2.0	83.0 ±4.4	58.0 ±1.4
KStar Diffuser (*num demos=50*)	81.0 ±3.6	94.3 ±2.5	24.3 ±3.1	27.7 ±3.2	85.0 ±2.7	62.5 ±1.4
KStar Diffuser (*num demos=100*)	**83.0** ±1.7	**98.7** ±1.5	**27.0** ±1.7	**43.7** ±4.5	**89.0** ±5.2	**68.2** ±2.1
Action Chunking Length						
KStar Diffuser (*chunking=1*)	**92.0** ±1.7	**98.7** ±0.6	23.7 ±5.9	16.0 ±5.2	12.3 ±4.2	48.5 ±0.6
KStar Diffuser (*chunking=2*)	83.0 ±1.7	**98.7** ±1.5	**27.0** ±1.7	**43.7** ±4.5	89.0 ±5.2	**68.2** ±2.1
KStar Diffuser (*chunking=5*)	81.3 ±1.5	97.7 ±2.5	1.7 ±1.2	14.3 ±3.2	**99.3** ±1.2	58.9 ±1.4
Historical Observation Length						
KStar Diffuser (*history=0*)	86.3 ±3.5	**99.7** ±0.6	9.7 ±2.1	0.0 ±0.0	0.0 ±0.0	39.1 ±1.1
KStar Diffuser (*history=1*)	**87.7** ±4.0	28.3 ±2.5	10.3 ±7.4	36.3 ±8.5	**92.7** ±1.5	51.1 ±1.5
KStar Diffuser (*history=2*)	83.0 ±1.7	98.7 ±1.5	**27.0** ±1.7	**43.7** ±4.5	89.0 ±5.2	**68.2** ±2.1
Coefficient λ						
KStar Diffuser (λ=*0.1*)	**84.3** ±4.9	**99.3** ±1.2	12.7 ±2.3	19.7 ±4.7	89.7 ±2.5	61.1 ±1.1
KStar Diffuser (λ=*0.5*)	83.7 ±11.6	98.3 ±1.5	3.3 ±1.2	36.3 ±6.8	**93.7** ±3.8	63.1 ±4.5
KStar Diffuser (λ=*0.9*)	83.0 ±1.7	98.7 ±1.5	**27.0** ±1.7	**43.7** ±4.5	89.0 ±5.2	**68.2** ±2.1

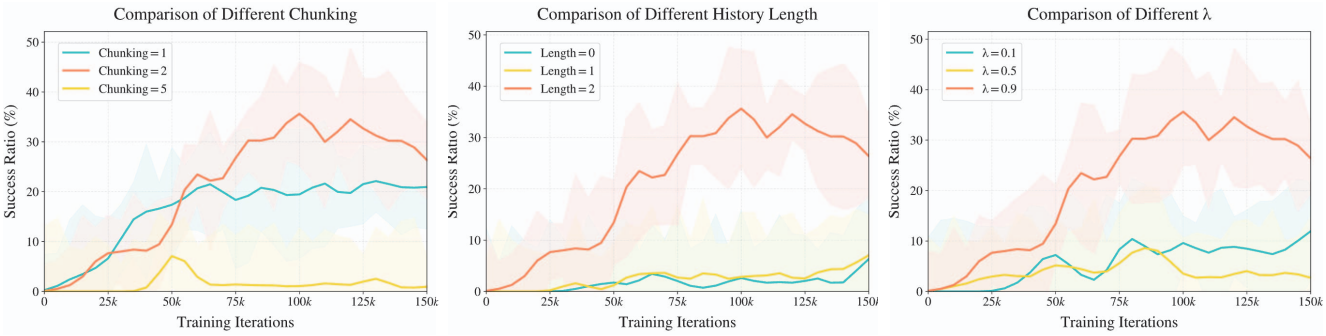


Figure C. *The left*: The result of different action chunking size. *The middle*: The result of history lengths. *The right*: The result of different coefficient λ.

erated (action chunk = 1), 2) two-step prediction of consecutive optimal poses (action chunk = 2), and 3) five-step prediction of sequential optimal poses (action chunk = 5).

The results presented in Table E reveal an **interesting trade-off between prediction horizon and model performance**. While single-step prediction (action chunk = 1) provides basic capabilities, it usually meets the multi-modal problem. The two-step prediction strategy (action chunk = 2) emerges as the optimal configuration, demonstrating superior success rates and motion quality across all tasks. Notably, attempting to predict longer sequences (action chunk = 5) leads to decreased performance, with success rates dropping by around 10% compared to the two-step configuration. This performance degradation suggests that when predicting next best pose, extended prediction horizons introduce excessive complexity into the learning problem, making it challenging for the policy to capture and generate accurate action sequences. As shown in Figure C, we present the variation in the success ratio for the `handover_item_easy` task, under different action chunking size configurations as the number of training steps increases.

(3) The Effects of Historical Observation Length. During our experiments, we found that the historical information plays an important role in action prediction quality. We conduct an ablation study across three history lengths: 0 (current observation only), 1, and 2 steps. The results

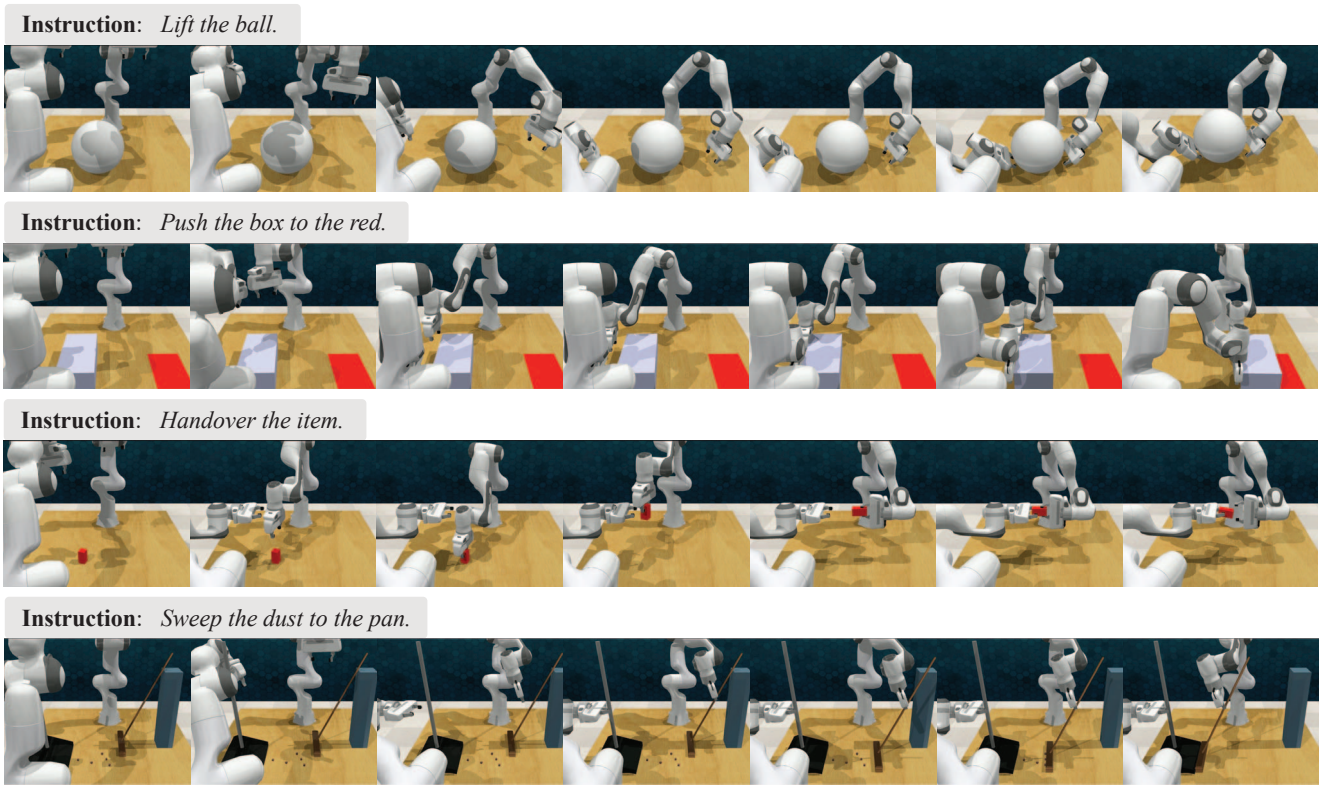


Figure D. The visualization of simulated tasks, including `push_box`, `lift_ball`, `handover_item_easy`, `sweep_dustpan`.

are shown in Table E. With no historical information (0-step), the policy fails to learn effective policies, as it cannot capture the temporal dependencies crucial for manipulation tasks. Adding one historical step enables basic learning capabilities, with the model achieving preliminary success in simpler tasks. Further extending to two historical steps yields optimal performance, showing an approximately 17% improvement over the single-step configuration and demonstrating enhanced stability across all tasks. While longer history lengths might provide more temporal context, they risk introducing redundant information that could potentially obscure relevant features, as observed in our preliminary experiments. These findings suggest that while temporal context is essential for understanding the current state and predicting actions, **an appropriate history window**, *e.g.*, 2 steps, provides an optimal balance between capturing necessary temporal dependencies and maintaining computational efficiency. As shown in Figure C, we present the variation in the success ratio for the `handover_item_easy` task, under different history length configurations as the number of training steps increases.

(4) The Effects of Coefficient λ. In policy learning, a kinematic regularization term is incorporated into the next best pose learning objective to balance task effectiveness and motion constrains. The regularization strength is controlled by the coefficient λ, which determines the extent of kinematic constraints imposed on the learning process. The choice of λ significantly influences policy performance. Larger values of λ (approaching 1.0) correspond to weaker kinematic constraints, enabling more flexible motion patterns. Conversely, smaller values of λ (e.g., 0.1) impose stricter kinematic constraints, yielding more conservative policies that emphasize motion smoothness over task efficiency. Empirical results, as shown in Table E, demonstrate that policy performance reaches its optimum at $\lambda = 0.9$. For smaller values of λ, the learned policies exhibit overly cautious behavior, manifesting in trajectories that prioritize smoothness at the expense of task efficiency and optimal path planning. Conversely, as λ approaches 1.0, the diminished kinematic constraints result in less regulated motion patterns, potentially compromising trajectory naturalness and precision. These findings indicate that $\lambda = 0.9$ achieves a better trade-off between preserving natural motion characteristics and ensuring efficient task execution. This configuration effectively minimizes the adverse effects of both excessive motion constraints and insufficient regulation, ultimately yielding superior performance across our task suite. As shown in Figure C, we present the variation in the success ratio for the `handover_item_easy` task,

under different coefficient λ configurations as the number of training steps increases.

D. Qualitative Analysis

We show more qualitative result in Figure D. Through the novel approach of encoding robotic arm structural information as graph representations and explicitly incorporating kinematic constraints, our model demonstrates exceptional performance in motion symmetry, synchronization, and coordination across dual-arm manipulation tasks. In the `lift_ball` experiment, the model achieves precise bilateral symmetry in spatial positioning through learned structured representations. The dual arms maintain stable symmetric configurations while preventing object instability through synchronized force application patterns, highlighting the efficacy of our structure-aware control paradigm. Furthermore, in the `push_box` task, the model exhibits remarkable geometric symmetry in motion planning and execution. By leveraging embedded kinematic information, the robotic arms consistently maintain equidistant positioning relative to the target object's center of mass while executing synchronized trajectories along parallel paths. This precise symmetrical control not only ensures operational stability during object manipulation but also establishes a robust framework for dual-arm cooperative control in complex manipulation scenarios.

E. Limitations and Future Directions.

While we explored robot structure impacts through GNN modeling and kinematic constraints, the core control logic of end-effector pose prediction and inverse kinematics remains. In the future, we aim to leverage neural networks to directly model joint movements, aligning the robot motion space with the Cartesian space of the human world.