

Learning to See and Act: Task-Aware Virtual View Exploration for Robotic Manipulation

Yongjie Bai^{1,2,*} Zhouxia Wang^{1,3,*} Yang Liu^{1,✉} Kaijun Luo¹ Yifan Wen¹ Mingtong Dai^{2,4}
Weixing Chen¹ Ziliang Chen² Lingbo Liu² Guanbin Li^{1,2} Liang Lin^{1,2,5}

¹Sun Yat-sen University ²Pengcheng Laboratory ³Nanyang Technological University

⁴Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences ⁵X-Era AI Lab

Appendix

In this supplementary material, we provide additional details and results of our proposed TVVE to complement the main paper. The content is organized as follows:

- **Appendix A:** Real-world experimental details. Corresponding **demo videos** of the results are also included in the supplementary materials.
- **Appendix B:** Additional design details of TVVE, including TaskMoE, point cloud aggregation, camera pose parameterization, and the proposed pseudo-environment interaction mechanism.
- **Appendix C:** More simulation experimental details and results.
- **Appendix D:** Implementation details and information about the RL Bench [7] and RL Bench-OG. Additionally, we provide more visualized results, with corresponding **demo videos** included in the supplementary materials.
- **Appendix E:** Multi-view re-rendering visualization results.

Appendix A. Real-World Experimental Details

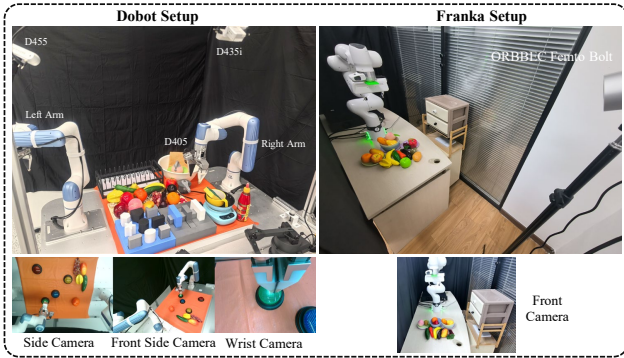


Figure 1. Real-World Environment Setup.

To validate the generalizability of the proposed TVVE

in real-world deployment scenarios, we conducted experiments on two distinct robotic platforms: the Dobot Nova 2 and the Franka Research 3, employing differentiated camera configurations (multi-view and single-view) to comprehensively validate the scenarios. For the Dobot Nova 2, a multi-view setup consisting of three perspectives (side, front, and wrist-mounted) was used, whereas for the Franka robot, only a front view was utilized. Experimental results confirm that our TVVE can be successfully deployed on both platforms and outperforms both Diffusion Policy and ARP. Figure 1 illustrates the real-world deployment environments and the corresponding images from each viewpoint.

Across five real-world tasks—*Pick Grape*, *Stack Bowls*, *Push Buttons*, *Collect Fruits*, and *Put Item In Drawer*—our TVVE achieves strong performance. Notably, in the *pick.grape* task, it successfully executed the manipulation even when the target was partially occluded in one viewpoint, demonstrating the advantage of our model in global perception capabilities. Corresponding demonstration videos are provided to facilitate a more intuitive understanding of the real-world outcomes.

Furthermore, to further evaluate the executive capability of our method across various types of tasks, we extended the experiments on the *Franka Research 3* platform to include ten additional tasks. These tasks encompass articulated object manipulation, deformable object manipulation, among others. Experimental results are summarized in Table 1, which indicate that our TVVE achieves high success rates, highlighting its adaptability and effectiveness across a diverse set of tasks.

Real-World Robustness and Generalization Testing.

We conduct robustness testing on the real-world *Pick Grape* task on the *Dobot nova 2* robot, evaluating under various conditions including unseen instances, unseen backgrounds, unseen objects, heavy occlusion, and illumination

* Equal Contribution ✉ Corresponding author

Arm	Method	Avg. SR (%) \uparrow	Pick Place	Stack Bowls	Push Buttons	Collect Fruits	Put Item In Drawer	Rotate Handle	Fold Towel	Open Lip	Unscrew Bottle	Reach Drag
Franka	ARP	62.0%	10/10	7/10	8/10	5/10	6/10	3/10	5/10	7/10	3/10	8/10
	TVVE (Ours)	70.0%	10/10	7/10	9/10	6/10	7/10	3/10	5/10	9/10	5/10	9/10

Table 1. Task success rates for Franka robots across more manipulation tasks.

Arm	Method	Avg. SR (%) \uparrow	Seen	Inst.	Bkg.	Obj.	Occl.	Illum.
Dobot	Diffusion Policy	53.3	90.0	80.0	70.0	60.0	10.0	10.0
	TVVE (Ours)	71.7	100.0	100.0	90.0	90.0	20.0	30.0

Table 2. Performance comparison under different configurations on the Dobot robot. Abbreviations: Inst. (Unseen Instance), Bkg. (Unseen Background), Obj. (Unseen Object), Occl. (Heavy Occlusion), Illum. (Illumination Variation).

variation. Throughout the experiments, heavy occlusion is identified as the primary cause of system failures. The results demonstrate that TVVE exhibits significantly better overall adaptability compared to the Diffusion Policy (DP) baseline. While both methods perform similarly in seen scenarios, TVVE substantially outperforms DP when confronted with unseen backgrounds, unseen objects, and especially under heavy occlusion. The success rates based on 10 trials for each configuration are reported, with quantitative comparisons detailed in Table 2.

Details of Real-World Tasks.

We design real-world tasks to encompass as diverse operation types as possible, including grasping and placing, pressing/pushing, articulated object manipulation, rotation, deformable object manipulation, fine manipulation, tool usage, etc. The design details of each task are elaborated below.

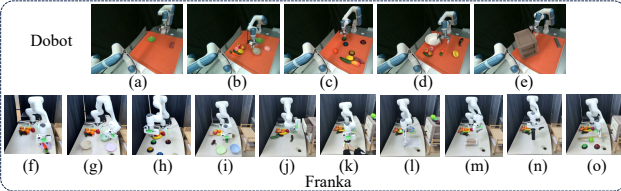


Figure 3. Real-World Tasks Setup.

Pick and Place. On the table, there are fruits and a plate. The robot needs to pick up and place the specified fruits. Language instruction: Put the [FRUIT] into the [COLOR] plate. As shown in Fig. 3 (a)(f).

Stack Bowls. There are some bowls and a plate on the table. The robot needs to stack the bowls and plate in a specific order. Language instruction: Stack a [COLOR] bowl and a [COLOR] bowl on a [COLOR] plate. As shown in Fig. 3 (b)(g).

Push Buttons. There are multiple buttons of different colors on the table. The robot needs to press the buttons in a specific sequence. Language instruction: Press the [COLOR] button, then the [COLOR] button, followed by the [COLOR] button, and finally the [COLOR] button. As shown in Fig. 3 (c)(h).

Collect Fruits. There are various fruits on the table. The robot needs to collect the specified fruits into the specific

target. Language instruction: Put a [FRUIT], a [FRUIT], and an [FRUIT] into the [COLOR] [TARGET]. As shown in Fig. 3 (d)(i).

Put Item In Drawer. There is a cabinet on the table. The robot needs to open the specified drawer, place the target object inside, and then close the drawer. Language instruction: Open the [Top/Middle/Bottom] drawer and put the [OBJECT] into it. As shown in Fig. 3 (e)(j).

Rotate Handle. The experimenter holds the handle, and the robot needs to rotate the handle by a certain angle. Language instruction: Rotate the handle clockwise. As shown in Fig. 3 (k).

Fold Towel. There is a towel on the table. The robot needs to pick up one corner of the towel and then fold it. Language instruction: Fold the towel. As shown in Fig. 3 (l).

Open Lip. There is a box on the table. The robot needs to pick up the lid and open it. Language instruction: Open the plastic lid of the biscuit case. As shown in Fig. 3 (m).

Unscrew Bottle. There is a bottle on the table. The robot needs to twist the cap to open the bottle. Language instruction: Unscrew the black bottle. As shown in Fig. 3 (n).

Reach Drag. There is a tool and an object to be manipulated on the table. The robot needs to use the tool to move the target object to the target position. Language instruction: Sweep the green apple into the gap between buttons using wood board. As shown in Fig. 3 (o).

Appendix B. More design details of TVVE

TaskMoE: Mathematical Principles and Operational Mechanism

To handle heterogeneous manipulation tasks, we introduce a task-aware Mixture-of-Experts (TaskMoE) that routes visual tokens through (i) a task-instruction-aware gate layer and (ii) a per-gate expert selector. Unlike prior MoE work [11, 13], TaskMoE decouples tasks from gates, decouples gates from experts, and adds semantic/entropy regularizers to avoid collapse.

Cross-modal conditioning and FiLM. Let $\mathbf{V} \in \mathbb{R}^{S \times D}$ denote the sequence of visual tokens, where S is the number of spatial tokens and D is the feature dimension. Let $\mathbf{I} \in \mathbb{R}^D$ be the language instruction embedding and $\mathbf{t} \in \mathbb{R}^D$ the learnable embedding of the current task. We first fuse language



Figure 2. Visualization of Dobot and Franka Robots' Execution Processes in Real-World Environments. In the figure, the T-axis represents the time axis, indicating the duration from the start to the end of the task. Each demonstration shows the following elements: side and wrist camera views for the Dobot robot, and a single front view for the Franka robot.

into vision via cross-modal attention:

$$\mathbf{Q} = \mathbf{V}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{I}\mathbf{W}^K, \quad \mathbf{V}_a = \mathbf{I}\mathbf{W}^V, \quad (1)$$

$$\mathbf{V}' = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V}_a, \quad (2)$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{D \times D}$ are projection matrices and d_k is the key dimension. The attention output $\mathbf{V}' \in \mathbb{R}^{S \times D}$ is an instruction-attended visual feature. We then modulate channels with FiLM using globally pooled vision and the task embedding (here $\text{GAP}(\cdot)$ is global average

pooling over the S tokens):

$$\mathbf{c} = [\text{GAP}(\mathbf{V}'); \mathbf{t}] \in \mathbb{R}^{2D}, \quad (3)$$

$$\gamma, \beta = \text{MLP}(\mathbf{c}) \in \mathbb{R}^D \times \mathbb{R}^D, \quad (4)$$

$$\mathbf{V}_{\text{mod}} = \gamma \odot \mathbf{V} + \beta, \quad (5)$$

where γ and β are per-channel scaling and bias vectors and \odot denotes element-wise multiplication. The modulated feature $\mathbf{V}_{\text{mod}} \in \mathbb{R}^{S \times D}$ serves as input to the gating network.

Stage 1: Task-aware gates ($N_G \ll N_J$). We consider N_J tasks and N_G latent gates. Gates represent reusable skill

clusters shared across tasks. A shared gate head scores N_G gates for each token and is modulated by a learnable task–gate assignment matrix $\mathbf{A} \in \mathbb{R}^{N_J \times N_G}$:

$$\mathbf{p}_g = \text{softmax}(\mathbf{V}_{\text{mod}} \mathbf{W}_g) \odot \text{softmax}(\mathbf{A}[j, :]), \quad (6)$$

where $\mathbf{W}_g \in \mathbb{R}^{D \times N_G}$ is the gate projection and $j \in \{1, \dots, N_J\}$ is the index of the current task. The matrix $\mathbf{p}_g \in \mathbb{R}^{S \times N_G}$ contains, for each token, a probability distribution over gates. We select the top-1 gate per token (keeping top-2 for analysis) and apply a per-gate capacity constraint to prevent any gate from absorbing too many tokens.

Stage 2: Per-gate expert selection (N_E arbitrary). Given the chosen gate $g \in \{1, \dots, N_G\}$, we select experts from an independent pool of N_E experts. Each gate has its own expert head:

$$\mathbf{z}_e = \mathbf{V}_{\text{mod}} \mathbf{W}_{\text{exp}}[g] + \mathbf{b}_{\text{exp}}[g], \quad \mathbf{p}_e = \text{softmax}(\mathbf{z}_e), \quad (7)$$

where $\mathbf{W}_{\text{exp}} \in \mathbb{R}^{D \times N_G \times N_E}$ and $\mathbf{b}_{\text{exp}} \in \mathbb{R}^{N_G \times N_E}$ are the gate–expert projection and bias, and $\mathbf{p}_e \in \mathbb{R}^{S \times N_E}$ gives, for each token, a probability distribution over experts under gate g . We keep top- k experts per token (default $k = 2$), optionally filter the second expert with a threshold/random policy, and enforce an expert capacity across the sequence. The resulting dispatch tensor aggregates these expert weights to route tokens to experts, and expert outputs are combined using the selected mixture weights.

Regularization and loss. Let $\mathcal{L}_{\text{task}}$ denote the main imitation learning loss. We add balance, entropy, and semantic regularizers:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_g \mathcal{L}_{\text{bal-gate}} + \lambda_e \mathcal{L}_{\text{bal-exp}} \quad (8)$$

$$- \eta_g H(\mathbf{p}_g) - \eta_e H(\mathbf{p}_e) + \lambda_{\text{sem}} \mathcal{L}_{\text{sem}}(\mathbf{A}, \mathbf{I}), \quad (9)$$

where $\mathcal{L}_{\text{bal-gate}}$ and $\mathcal{L}_{\text{bal-exp}}$ penalize unbalanced gate/expert utilization, $H(\mathbf{p}_g)$ and $H(\mathbf{p}_e)$ are the entropies of the gate and expert distributions (encouraging diverse routing), and $\mathcal{L}_{\text{sem}}(\mathbf{A}, \mathbf{I})$ aligns rows of the task–gate matrix \mathbf{A} with the similarity structure of instruction embeddings \mathbf{I} so that semantically related tasks share gates/experts. The scalars $\lambda_g, \lambda_e, \eta_g, \eta_e, \lambda_{\text{sem}} \geq 0$ control the strength of each regularizer. Capacity clipping at both stages further prevents any single gate or expert from monopolizing the traffic.

Scalability and specialization. Because gates are decoupled from tasks and from experts, we can set the number of gates N_G and experts N_E independently (e.g., $N_G=8$, $N_E=16$). The gate layer captures coarse task affinity (each task uses only a few gates), while the expert layer refines routing inside each gate into task-specific specialists. Empirically, increasing N_E leads to finer skill factorization without leaving experts idle, validating the two-stage gate→expert design with entropy/semantic regularization.

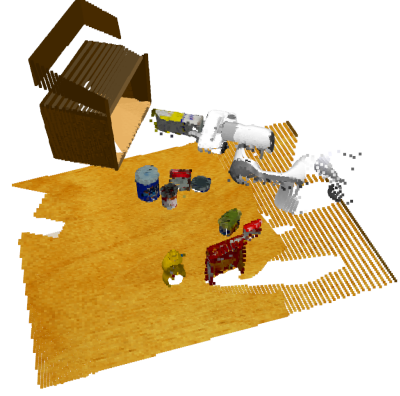


Figure 4. The aggregated full-scene point cloud of the *put_groceries_in_cupboard* task from multiple viewpoints

Point Cloud Aggregation.

In the RL Bench dataset experiments, we select RGB-D images from four fixed viewpoints: *front*, *right_shoulder*, *left_shoulder*, and *wrist* as inputs. Following the approach of RVT-2 [5], we convert the depth maps from all viewpoints into point clouds in a global coordinate system using the intrinsic and extrinsic camera parameters. Subsequently, we perform multi-view point cloud aggregation and remove redundant points outside the workspace to obtain the final scene point cloud, as illustrated in Fig. 4.

Pseudo-environment Interaction Mechanism.

To enable efficient policy optimization without physical environment interaction, we design a novel *pseudo-environment* interaction mechanism. As shown in Algorithm 1. The shadow network π_{shadow} (a frozen copy of the policy before training) processes observation \mathbf{o} to compute reference losses \mathcal{L}_{ref} . MVEP π_{θ} processes \mathbf{o} to generate camera poses \mathbf{p} (\mathbf{p} as policy action), the rendered observation images from multi-view camera poses are fed into TaskMoE and TaskMoE-ARP (Autoregressive Action Policy) to compute the current policy losses $\mathcal{L}_{\text{TVVE}}$. The reward r is then calculated based on \mathcal{L}_{ref} , $\mathcal{L}_{\text{TVVE}}$, and \mathbf{p} . Finally, the experience tuple $(\mathbf{o}, r, \mathbf{p}, \log \pi_{\theta_{\text{old}}}(\mathbf{p}|\mathbf{o}), V_{\theta_{\text{old}}}(\mathbf{o}))$ is stored in the replay buffer \mathcal{B} for policy updates. This approach enables batch environment interaction, leveraging existing demonstration data to improve policies and enhance data utilization efficiency.

Camera Pose Parameterization.

The camera pose is mathematically represented using a **look-at model** defined by a 5-dimensional parameter vector $\mathbf{p}^i = (\theta^i, \phi^i, r^i, \theta_{\text{up}}^i, \phi_{\text{up}}^i) \in \mathbb{R}^5 (i \in [0, K-1], K \text{ is number of camera viewpoint})$. This representation decouples camera position from orientation through spherical coordinates:

Algorithm 1 Pseudo-Environment Interaction

- 1: **Input:** Current observation \mathbf{o} , shadow network π_{shadow}
 - 2: **Output:** Transition tuple τ
 - 3: # Compute reference loss
 - 4: $\mathcal{L}_{\text{ref}} \leftarrow \pi_{\text{shadow}}(\mathbf{o})$
 - 5: # Generate camera poses & compute current model loss using TVVE
 - 6: $\mathcal{L}_{\text{TVVE}}, \mathbf{p}, \log \pi_{\theta_{\text{old}}}(\mathbf{p}|\mathbf{o}), V_{\theta_{\text{old}}}(\mathbf{o}) \leftarrow \pi_{\theta_{\text{old}}}(\mathbf{o})$
 - 7: # Compute reward
 - 8: $r \leftarrow \text{RewardCalculator}(\mathcal{L}_{\text{ref}}, \mathcal{L}_{\text{TVVE}}, \mathbf{p})$
 - 9: # Construct transition tuple
 - 10: **return** $(\mathbf{o}, \mathbf{p}, r, \log \pi_{\theta_{\text{old}}}(\mathbf{p}|\mathbf{o}), V_{\theta_{\text{old}}}(\mathbf{o}))$
-

$$\mathbf{t}^i = \begin{bmatrix} x^i \\ y^i \\ z^i \end{bmatrix} = \begin{bmatrix} r^i \sin \theta^i \cos \phi^i \\ r^i \sin \theta^i \sin \phi^i \\ r^i \cos \theta^i \end{bmatrix} \quad (10)$$

where \mathbf{t}^i denotes the camera center position. The viewing direction is intrinsically defined as $\mathbf{v}_{\text{look}}^i = -\mathbf{t}^i / \|\mathbf{t}^i\|_2$ toward the **origin**. The up-vector orientation is parameterized as:

$$\mathbf{v}_{\text{up}}^i = \begin{bmatrix} \sin \theta_{\text{up}}^i \cos \phi_{\text{up}}^i \\ \sin \theta_{\text{up}}^i \sin \phi_{\text{up}}^i \\ \cos \theta_{\text{up}}^i \end{bmatrix} \quad (11)$$

The final camera matrix is constructed via Gram-Schmidt orthogonalization between $\mathbf{v}_{\text{look}}^i$ and \mathbf{v}_{up}^i .

Appendix C. More Experimental Details and Results

Baseline Details.

We compare with eleven state-of-the-art baselines on RL-Bench, RLbench-OG and Real-world, including both 2D and 3D approaches. For 2D methods, we include **Diffusion Policy** [2], which formulates robot policies as conditional denoising diffusion processes. The 3D baselines comprise voxel-based methods: **C2F-ARM-BC** [8], which predicts actions in a coarse-to-fine manner via Q-value estimation; **PerAct** [12], which detects actions through global self-attention; **Hiveformer** [6], which employs attention across historical features; and **GNFactor** [14], which co-optimizes a neural scene representation with a PerAct-based policy. We also consider point-based methods: **PolarNet** [1], which computes dense point representations, and multi-view approaches: **RVT** [4], which fuses multi-view predictions via 3D back-projection, and its successor **RVT2** [5], which enhances precision and efficiency. Furthermore, we

include **Act3D** [3], which uses coarse-to-fine 3D featurization (retrained by us for a fair comparison), **3D Diffuser Actor** [9], a 3D diffusion-based policy, and **ARP** [15], an autoregressive policy generating hybrid action sequences via a Chunking Causal Transformer.

Implementation Details.

In the RLbench *Multi-view setup* experiments, during the supervised pre-training phase, our experimental hyperparameter settings comply with APR[15]. In the offline reinforcement learning phase for MVEP, our hyperparameter configurations are detailed in Table 3.

Hyperparameter	Value
<i>MVEP</i>	
K number of the camera viewpoints	3
N_{MVEP} The number of input point clouds for MVEP	2048
embedding size	512
<i>Train & Eval</i>	
observation(RGBD)	$4 \times 128 \times 128 \times 4$
re-render image resolution	224×224
maximum evaluation steps	25
train epochs	20
eval frequency	100
batch size	96
learning rate	2.0e-6
learning rate scheduler	cosine
optimizer	LAMB

Table 3. Hyperparameters for offline reinforcement learning training in RLbench experiments.

Performance-Efficiency Trade-off Analysis.

During inference, TVVE must predict dynamic multi-view camera poses and continuously adjust the rendering camera pose, resulting in a non-negligible computational overhead. To assess the practicality of the proposed TVVE model, we compare its average task success rate and inference latency with those of the baseline model, ARP. The results, summarized in Table 4 on the RLbench, show that TVVE achieves a higher average success rate than ARP, while the increase in inference latency remains relatively modest (approximately 10.7%). These findings suggest that TVVE effectively balances task performance and operational efficiency, with design elements such as sampling acceleration and camera caching playing a key role in enhancing model efficiency.

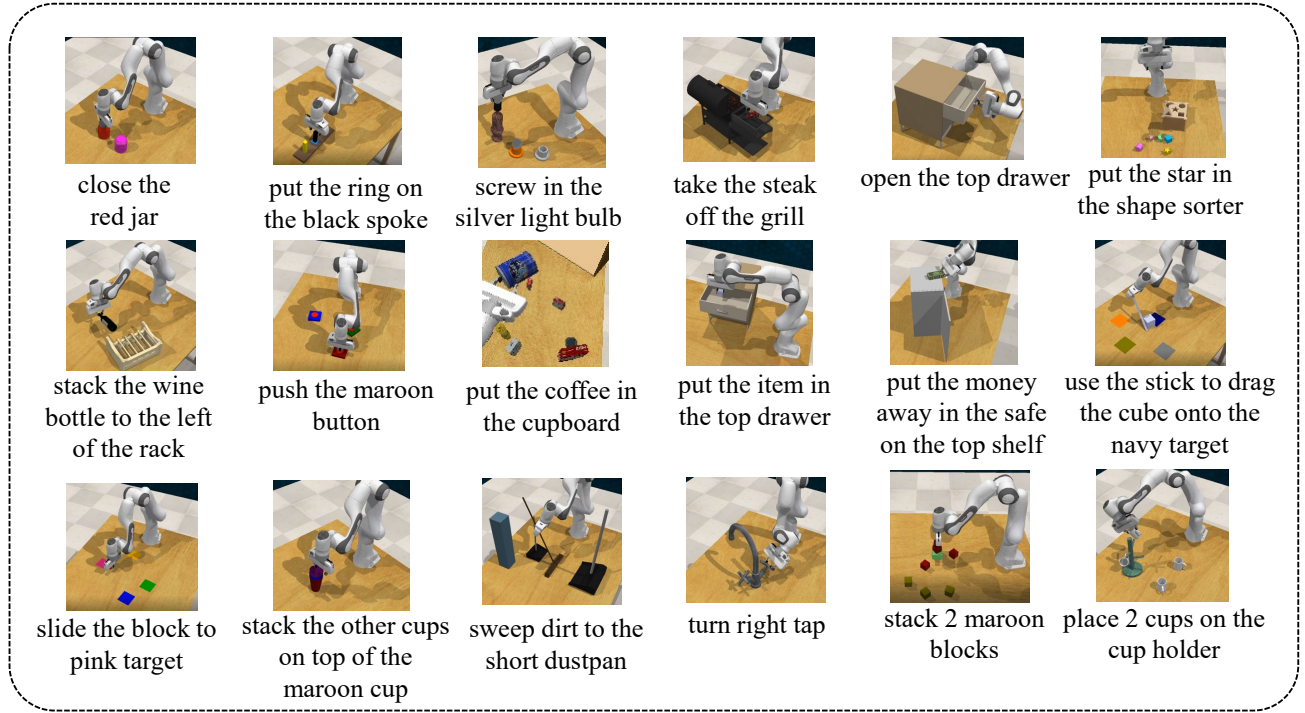


Figure 5. Demonstrations of 18 RL Bench tasks and their corresponding language instructions.

Method	Average Success Rate (%)	Average Inference Time(s)
ARP	81.6	0.394
TVVE	86.6	0.436

Table 4. Comparison of the average success rate and inference time between ARP and TVVE across 18 tasks.

Method	20	40	80	100
ARP	6.7 ± 2.3	14.7 ± 2.3	28.0 ± 4.0	45.3 ± 6.1
TVVE (Ours)	10.7 ± 2.3	17.3 ± 2.3	30.7 ± 6.1	52.0 ± 4.0

Table 5. Success rate (mean \pm std %) comparison under varying numbers of demonstrations on the RL Bench *Put in Cupboard* task.

Data Efficiency Analysis.

We conduct a demonstration ablation study on the RL Bench *Put in Cupboard* task. As shown in Table 5, TVVE consistently outperforms the baseline ARP across all demonstration sizes (20, 40, 80, and 100). Notably, the performance gap widens as the number of demonstrations increases, with TVVE achieving a success rate of $52.0 \pm 4.0\%$ compared to $45.3 \pm 6.1\%$ for ARP at 100 demonstrations. This trend demonstrates that while more data benefits both methods, the superior sample efficiency and final performance of TVVE are inherently due to our proposed architectural improvements, not simply the result of scaling the dataset.

Results on RL Bench-OG.

We conducted an evaluation of TVVE, RVT2, ARP, and Diffusion Policy on the RL Bench-OG benchmark. Detailed performance across 10 tasks under 8 variations is presented in Tables 6, 7, 8, and 9. The reported success rates correspond to the mean \pm standard deviation from three independent trials.

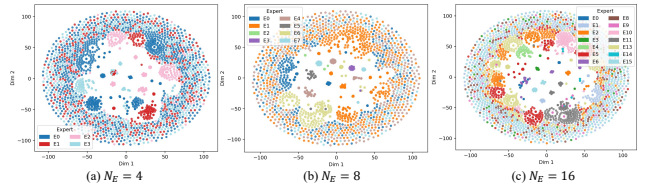


Figure 6. t-SNE visualization of instruction embeddings routed through the 4-expert (a), 8-expert (b) and 16-expert (c) TaskMoE. In the 4-expert TaskMoE (a), each expert bears a high workload, their specialization is limited, and it is difficult to achieve fine-grained partitioning. In the 8-expert TaskMoE (b), inner-region experts (E0, E1, E2, E3, E4, E5, E6, E7) form compact, semantically coherent clusters, while outer-ring embeddings are dominated by E1, E4 and E7, indicating fallback routing for diverse or weakly structured instructions. Compared to the 8-expert configuration, the 16-expert TaskMoE exhibits more balanced expert utilization. N_E denotes the number of experts.

Experimental Results and Analysis on TaskMoE.

In the RL Bench *Multi-view setup* experiments, we conducted individual ablation studies on the components of TaskMoE, and visualized and analyzed the routing behavior of the gate-expert at both the **instruction-level** and **task-level**.

Instruction-level Routing Behavior Analysis. We compare the routing behavior of the 16-expert Task-MoE with the 4-expert and 8-expert configuration using instruction-level t-SNE embeddings, as shown in Fig. 6. The 16-expert model exhibits markedly improved semantic disentanglement: the embedding manifold contains a substan-

Task Name	Occlusion 1	Occlusion 2	Light Color	Table Color	Table Texture	Distractor	Background Texture	Camera Pose	Variant Mean
basketball_in_hoop	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	92.0 \pm 0.0	96.0 \pm 0.0	70.0 \pm 2.0	100.0 \pm 0.0	100.0 \pm 0.0	94.8 \pm 9.8
scoop_with_spatula	88.0 \pm 8.0	88.0 \pm 5.7	64.0 \pm 0.0	76.0 \pm 0.0	86.0 \pm 2.0	68.0 \pm 4.0	92.0 \pm 4.0	89.3 \pm 3.8	81.4 \pm 10.8
take_plate_off_colored_dish_rack	100.0 \pm 0.0	98.7 \pm 1.9	76.0 \pm 4.0	94.0 \pm 6.0	92.0 \pm 0.0	100.0 \pm 0.0	98.7 \pm 1.9	100.0 \pm 0.0	94.9 \pm 8.2
water_plants	88.0 \pm 4.0	26.7 \pm 3.8	16.0 \pm 4.0	16.0 \pm 0.0	20.0 \pm 4.0	22.7 \pm 1.9	24.0 \pm 4.0	24.0 \pm 3.3	29.7 \pm 22.6
block_pyramid	8.0 \pm 8.0	0.0 \pm 0.0	22.0 \pm 6.0	8.0 \pm 0.0	8.0 \pm 0.0	12.0 \pm 0.0	16.0 \pm 4.0	18.0 \pm 6.0	11.5 \pm 7.9
solve_puzzle	2.0 \pm 2.0	0.0 \pm 0.0	14.0 \pm 6.0	14.0 \pm 2.0	10.0 \pm 2.0	12.0 \pm 0.0	24.0 \pm 0.0	22.0 \pm 2.0	12.2 \pm 8.3
take_usb_out_of_computer	100.0 \pm 0.0	100.0 \pm 0.0	97.3 \pm 1.9	100.0 \pm 0.0	96.0 \pm 0.0	82.7 \pm 1.9	100.0 \pm 0.0	100.0 \pm 0.0	97.0 \pm 5.7
close_drawer	100.0 \pm 0.0	100.0 \pm 0.0	92.0 \pm 0.0	96.0 \pm 0.0	96.0 \pm 0.0	98.7 \pm 1.9	98.7 \pm 1.9	98.7 \pm 1.9	97.5 \pm 2.8
straighten_ropes	64.0 \pm 0.0	0.0 \pm 0.0	58.0 \pm 2.0	50.0 \pm 2.0	64.0 \pm 8.0	36.0 \pm 0.0	89.3 \pm 5.0	80.0 \pm 4.0	55.2 \pm 26.2
toilet_seat_down	100.0 \pm 0.0	66.7 \pm 8.2	97.3 \pm 1.9	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	95.5 \pm 11.3
Task Mean	75.0 \pm 1.2	58.0 \pm 1.1	63.7 \pm 1.1	64.6 \pm 0.7	66.8 \pm 0.9	60.2 \pm 0.6	74.3 \pm 0.9	73.2 \pm 0.9	67.0 \pm 6.2

Table 6. Success Rates of TVVE under Different Perturbations of RL-Bench-OG.

Task Name	Occlusion 1	Occlusion 2	Light Color	Table Color	Table Texture	Distractor	Background Texture	Camera Pose	Variant Mean
basketball_in_hoop	100.0 \pm 0.0	100.0 \pm 0.0	88.8 \pm 1.6	96.0 \pm 0.0	99.2 \pm 1.6	88.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	96.5 \pm 4.8
scoop_with_spatula	82.7 \pm 7.5	84.0 \pm 6.5	75.4 \pm 2.6	84.0 \pm 5.7	88.0 \pm 5.7	76.0 \pm 4.0	98.0 \pm 2.0	96.0 \pm 0.0	85.5 \pm 7.7
take_plate_off_colored_dish_rack	98.7 \pm 1.9	65.3 \pm 6.8	60.0 \pm 4.6	88.8 \pm 3.9	84.8 \pm 4.7	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	87.2 \pm 15.2
water_plants	72.0 \pm 8.6	20.0 \pm 5.7	12.0 \pm 5.1	9.6 \pm 9.3	23.0 \pm 5.9	22.0 \pm 2.0	26.0 \pm 2.0	32.0 \pm 0.0	27.1 \pm 18.3
block_pyramid	4.0 \pm 3.3	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.5 \pm 1.3
solve_puzzle	0.0 \pm 0.0	0.0 \pm 0.0	9.3 \pm 3.0	8.0 \pm 3.6	12.0 \pm 5.1	20.0 \pm 0.0	16.0 \pm 3.3	25.3 \pm 3.8	11.3 \pm 8.4
take_usb_out_of_computer	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	95.2 \pm 1.6	98.0 \pm 2.0	98.7 \pm 1.9	100.0 \pm 0.0	99.0 \pm 1.6
close_drawer	100.0 \pm 0.0	100.0 \pm 0.0	99.3 \pm 1.5	97.6 \pm 2.0	98.4 \pm 2.0	96.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	98.9 \pm 1.4
straighten_ropes	70.7 \pm 10.0	0.0 \pm 0.0	56.7 \pm 5.8	37.6 \pm 5.4	44.0 \pm 5.1	36.0 \pm 0.0	86.7 \pm 5.0	88.0 \pm 4.0	52.5 \pm 27.6
toilet_seat_down	100.0 \pm 0.0	0.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	98.0 \pm 2.0	100.0 \pm 0.0	100.0 \pm 0.0	87.2 \pm 33.0
Task Mean	72.8 \pm 2.0	46.9 \pm 0.4	60.8 \pm 1.2	61.8 \pm 1.1	64.0 \pm 2.0	63.4 \pm 0.2	72.6 \pm 0.6	74.0 \pm 0.8	64.5 \pm 8.3

Table 7. Success Rates of RVT2 under Different Perturbations of RL-Bench-OG.

tially larger number of compact and well-separated clusters, each corresponding to an expert with strong specialization. Expert utilization becomes significantly more balanced, with reduced reliance on fallback experts for long-tail or weakly structured instructions. The outer-ring region—typically occupied by generic or ambiguous instructions—is shared by a diverse subset of experts rather than collapsing onto one or two dominant experts. This indicates that the expanded expert capacity enables finer-grained semantic partitioning and mitigates routing collapse. In contrast, the 8-expert model shows partially collapsed routing behavior. Although several inner-region experts still form coherent semantic clusters, the outer ring is dominated by E1, E4 and E7, which assume the role of catch-all experts. Their dispersed and high-density presence suggests that the gate struggles to allocate long-tail and noisy instructions across multiple experts. Consequently, the semantic boundaries between experts remain coarse, limiting the degree of modularization and reducing the benefits of MoE specialization. Overall, the comparison demonstrates that increasing the number of experts from 8 to 16 leads to more stable routing, finer semantic granularity, and healthier expert balancing, thereby enabling a more expressive and disentangled mixture-of-experts structure.

Task-level Routing Behavior Analysis. We analyze the gate and expert routing behavior by visualizing the task-wise distributions. Such visualization allows us to examine whether different RL-Bench tasks activate distinct subsets of gates and experts, revealing the degree of task specialization.

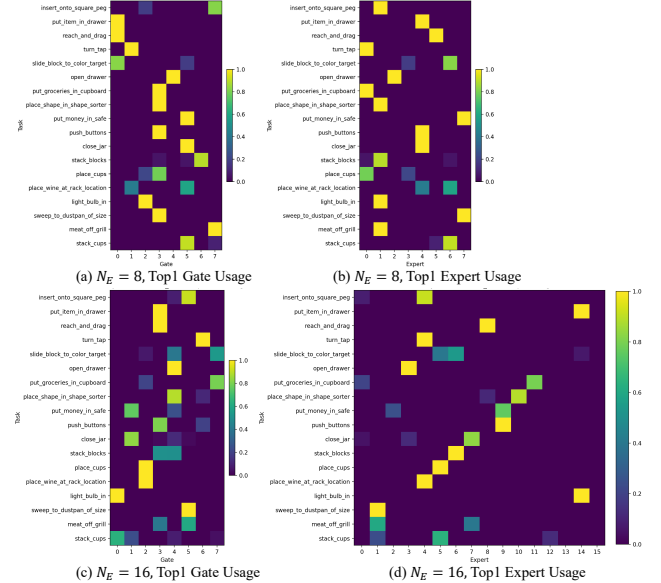


Figure 7. Task-wise gate and expert usage visualization for the proposed MoE layer. Both 8-expert (a,b) and 16-expert (c,d) configurations demonstrate clear task-dependent routing, while larger expert capacity allows finer-grained specialization without leading to expert collapse.

tion, routing stability, and expert utilization diversity within the TaskMoE architecture.

Fig. 7 shows per-task top-1 gate/expert usage for $N_E = 8$ (a,b) and $N_E = 16$ (c,d). With 8 experts, every task settles on 1–2 gates (e.g., *place_shape_in_shape_sorter* and *put_groceries_in_cupboard* both enter G3; *place_cups* enters G2/G3), and all gates/experts are exercised, confirming the

Task Name	Occlusion 1	Occlusion 2	Light Color	Table Color	Table Texture	Distractor	Background Texture	Camera Pose	Variant Mean
basketball_in_hoop	100.0 \pm 0.0	100.0 \pm 0.0	86.0 \pm 2.0	76.0 \pm 0.0	86.0 \pm 2.0	84.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	91.5 \pm 9.0
scoop_with_spatula	88.0 \pm 4.0	76.0 \pm 8.0	62.0 \pm 2.0	78.0 \pm 2.0	86.0 \pm 2.0	82.0 \pm 2.0	86.7 \pm 10.0	84.0 \pm 8.0	80.3 \pm 9.8
take_plate_off_colored_dish_rack	100.0 \pm 0.0	76.0 \pm 4.0	74.0 \pm 6.0	96.0 \pm 0.0	94.0 \pm 6.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	92.5 \pm 10.9
water_plants	66.0 \pm 14.0	10.0 \pm 2.0	10.0 \pm 2.0	12.0 \pm 0.0	10.7 \pm 3.8	13.3 \pm 7.5	16.0 \pm 8.0	21.3 \pm 3.8	19.9 \pm 19.0
block_pyramid	14.0 \pm 2.0	0.0 \pm 0.0	1.0 \pm 1.7	5.3 \pm 3.8	0.8 \pm 1.6	2.4 \pm 3.2	2.0 \pm 3.5	6.4 \pm 8.2	4.0 \pm 5.7
solve_puzzle	0.0 \pm 0.0	0.0 \pm 0.0	14.0 \pm 2.0	8.0 \pm 0.0	6.0 \pm 2.0	16.0 \pm 8.6	8.0 \pm 4.0	14.0 \pm 6.0	8.2 \pm 7.1
take_usb_out_of_computer	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	98.0 \pm 2.0	96.0 \pm 0.0	94.0 \pm 2.0	100.0 \pm 0.0	98.7 \pm 1.9	98.3 \pm 2.4
close_drawer	100.0 \pm 0.0	100.0 \pm 0.0	96.0 \pm 0.0	98.0 \pm 2.0	100.0 \pm 0.0	94.0 \pm 2.0	100.0 \pm 0.0	100.0 \pm 0.0	98.5 \pm 2.4
straighten_rope	62.0 \pm 6.0	4.0 \pm 0.0	60.0 \pm 0.0	60.0 \pm 0.0	46.0 \pm 6.0	42.0 \pm 6.0	68.0 \pm 8.0	74.0 \pm 2.0	52.0 \pm 21.2
toilet_seat_down	100.0 \pm 0.0	60.0 \pm 4.0	94.7 \pm 1.9	96.0 \pm 0.0	88.0 \pm 0.0	96.0 \pm 0.0	100.0 \pm 0.0	98.7 \pm 1.9	91.7 \pm 12.6
Task Mean	73.0 \pm 1.6	52.6 \pm 1.0	59.8 \pm 0.8	62.7 \pm 0.5	61.3 \pm 1.0	62.4 \pm 1.4	68.1 \pm 1.6	69.7 \pm 1.4	63.7 \pm 6.1

Table 8. Success Rates of ARP under Different Perturbations of RL Bench-OG.

Task Name	Occlusion 1	Occlusion 2	Light Color	Table Color	Table Texture	Distractor	Background Texture	Camera Pose	Variant Mean
basketball_in_hoop	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0
scoop_with_spatula	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	1.3 \pm 1.9	1.3 \pm 1.9	0.0 \pm 0.0	0.3 \pm 1.1
take_plate_off_colored_dish_rack	4.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	2.0 \pm 2.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.8 \pm 1.6
water_plants	2.0 \pm 2.0	4.0 \pm 0.0	8.0 \pm 0.0	4.0 \pm 3.3	2.0 \pm 2.0	8.0 \pm 3.3	2.0 \pm 2.0	2.0 \pm 2.0	4.0 \pm 3.3
block_pyramid	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0
solve_puzzle	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0
take_usb_out_of_computer	92.0 \pm 8.0	82.0 \pm 2.0	61.3 \pm 3.8	58.7 \pm 5.0	66.0 \pm 2.0	76.0 \pm 0.0	65.3 \pm 7.5	64.0 \pm 12.0	70.7 \pm 12.5
close_drawer	76.0 \pm 0.0	56.0 \pm 8.0	86.7 \pm 5.0	88.0 \pm 3.3	72.0 \pm 0.0	84.0 \pm 3.3	92.0 \pm 3.3	74.0 \pm 2.0	78.6 \pm 11.6
straighten_rope	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0
toilet_seat_down	100.0 \pm 0.0	92.0 \pm 8.0	73.3 \pm 5.0	74.7 \pm 1.9	84.0 \pm 4.0	74.7 \pm 13.2	88.0 \pm 8.6	82.0 \pm 2.0	83.6 \pm 11.1
Task Mean	27.4 \pm 0.8	23.4 \pm 1.1	22.9 \pm 0.8	22.5 \pm 0.7	22.6 \pm 0.5	24.4 \pm 1.4	24.9 \pm 1.2	22.2 \pm 1.2	23.8 \pm 1.9

Table 9. Success Rates of Diffusion Policy under Different Perturbations of RL Bench-OG.

task-aware gate design learns a stable task \rightarrow gate partition without idle capacity. Increasing to 16 experts keeps the gate partition similar but further splits the downstream experts: tasks that shared a single expert at $N_E = 8$ now route to distinct experts (e.g., *meat_off_grill* disperses over E1/E7 (from E1), *place_shape_in_shape_sorter* shifts to E8/E10 (from E1), *put_money_in_safe* to E2/E9 (from E7), yielding finer skill specialization. This validates that the two-stage gate \rightarrow expert routing, combined with entropy/semantic regularization and a larger N_E , not only provides additional capacity that is fully utilized without collapsing to a few experts but also enhances sparsity, which optimizes computational efficiency and improves generalization by distributing representations more distinctly.

Moreover, Fig. 7(c,d) shows the two-stage router differentiates fine-grained skills even when tasks share the same gate. For example, the semantically related tasks *place_cups* and *place_wine_at_rack_location* both enter the same gate (G2) at $N_E = 16$, yet their traffic is split to different experts downstream (e.g., *place_cups* to E5 vs. *place_wine_at_rack_location* to E4). This demonstrates that the gate layer captures coarse task affinity while the expert layer refines the routing to task-specific specialists, confirming the effectiveness of the two-stage gate \rightarrow expert design in achieving fine-grained routing without losing shared structure.

Generalization. We conduct experiments in RL Bench, training on tasks such as *Insert Peg* and *Put Item Drawer*, and others, and is subsequently evaluated on a set of unseen

tasks. Our model achieves high success rates in completing certain tasks not encountered during training, demonstrating the generalization capability of our architecture. The results are shown in Table 10.

Appendix D. Implementation details and information about the RL Bench and RL Bench-OG RL Bench.

To evaluate the effectiveness and generalization capability of the proposed TVVE for multi-task robotic manipulation, we conduct experiments on 18 diverse tasks from the RL Bench benchmark, including *close jar*, *stack blocks*, and so on. TVVE is trained with 100 demonstrations per task. Example demonstration samples for each task are shown in Fig. 5. During evaluation, the model is tested with 25 demonstrations using the same settings. For a more intuitive understanding, we also include corresponding video results.

RL Bench-OG: Task Definition and Visualization.

RL Bench-OG is derived from the RL Bench benchmark and is designed to evaluate the robustness of models under occlusion as well as their generalization capability under various environmental perturbations. RL Bench-OG selects ten tasks from the original RL Bench task list, covering both simple scenarios (e.g., *take_usb_out_of_computer*) and more complex long-horizon tasks (e.g., *block_pyramid*). The benchmark consists of two components: an **Occlusion Suite** and a **Generalization Suite**. We detail both compo-

Avg. SR (%)	Seen																	Unseen			
	Insert onto Square Peg in Drawer	Put Item in Drawer	Reach and Drag	Turn Tap	Slide Block to Color Target	Open Drawer	Put Groceries in Cupboard	Place Shape in Shape Sorter	Put Money in Safe	Push Buttons	Close Jar	Stack Blocks	Place Cups	Place Wine at Rack Location	Light Bulb In	Sweep to Dustpan of Size	Meat off Grill	Stack Cups	Water Plants	Close Drawer	Toilet Seat Down
80.2	24.0	100.0	96.0	100.0	100.0	88.0	68.0	32.0	96.0	96.0	100.0	68.0	36.0	96.0	88.0	96.0	96.0	64.0	12.0	44.0	100.0

Table 10. **Performance with $N_G = 8, N_E = 16$ TaskMoE in RL Bench.** TaskMoE enhances generalization to unseen tasks.

nents below. For the visualization of different variant settings corresponding to each task, see Fig. 8 and Fig. 9.

Occlusion Suite

Occlusion refers to situations in which the line of sight of the camera to key task-relevant regions is fully or partially blocked, leading to incomplete observations and degraded performance of state estimation and action execution. In the occlusion suite, we introduce occlusions to the *front_camera* through two mechanisms:

1. **Self-occlusion by object pose perturbation.** We modify the position or orientation of task-relevant objects such that essential interaction points become occluded. These occluded regions are often critical for task completion, such as the drawer handle in the *close_drawer* task.
2. **Occlusion by external distractors.** We place task-irrelevant objects—such as cabinets, TVs, or doors—in front of the workspace to partially block the scene, leading to incomplete visibility of key regions.

Task Construction. The following describes how occlusions are introduced for each of the ten tasks:

- **basketball_in_hoop:** Basket and trash can poses are perturbed to occlude the basketball.
- **block_pyramid:** A cabinet is placed in front of the workspace to occlude part of the blocks.
- **close_drawer:** The drawer is rotated such that its geometry occludes the handle.
- **scoop_with_spatula:** A wine bottle is positioned to block the target cube.
- **solve_puzzle:** A storage cabinet is placed to occlude puzzle pieces.
- **straighten_rope:** A desk lamp is placed in front of one end of the rope.
- **take_plate_off_colored_dish_rack:** A box with a laptop blocks visibility of the plate.
- **take_usb_out_of_computer:** A cabinet blocks the USB port area.
- **toilet_seat_down:** A door is placed such that it occludes the toilet seat.
- **water_plants:** A television partially blocks both the watering can and the plant.

Experimental Settings. We evaluate models under two occlusion levels, **Occlusion 1** and **Occlusion 2**. For **Occlusion 1**, models are both trained and tested directly under the occluded task configurations; for **Occlusion 2**, models

are trained in the original RL Bench task settings and then evaluated in a zero-shot manner under occluded conditions.

Generalization Suite

The Generalization Suite evaluates robustness to environment-conditioned variations. Based on the same ten tasks, we construct six types of environment variations, each modifying exactly one factor while keeping all others unchanged. Following the pipeline from the COLOSSEUM [10], we specify variation types using *yaml* configuration files and data collection procedures via *json* metadata.

Variation Types.

- **light_color:** RGB values are sampled within predefined ranges and applied to directional lights.
- **table_texture:** A texture is sampled from a texture dataset and applied to the table.
- **table_color:** RGB values are sampled within predefined ranges and applied to the table surface.
- **background_texture:** A background texture is randomly sampled and applied.
- **distractor:** Two distractor objects are sampled from a 3D asset dataset and spawned within the workspace boundary.
- **camera_pose:** Camera position and orientation offsets are sampled and applied to the front, left-shoulder, and right-shoulder cameras.

Experimental Settings. For the Generalization Suite, models are trained in the original RL Bench environment and subsequently evaluated in a zero-shot manner across various generalization variants. For all variants, we collect 25 validation episodes for each task.

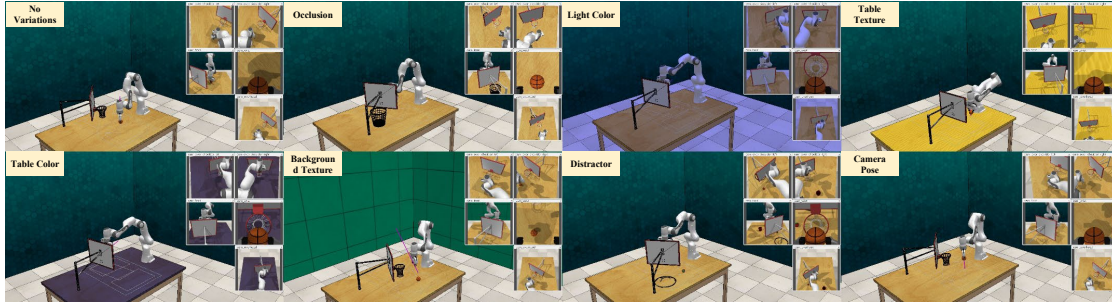
Appendix E. Multi-view Re-rendering Visualization Results

To further illustrate the operational principles of our TVVE framework and its performance on RL Bench, we visualize multiple dynamic virtual viewpoints generated by MVEP in 3D space at intermediate execution steps during inference, along with their corresponding rendered 2D images for each scene, as shown in Fig. 10 and Fig. 11. The rendered imagery clearly captures both the end-effector and target objects, enabling the action model to make more precise motion predictions based on these perceptual cues. This enhancement directly contributes to improved task success

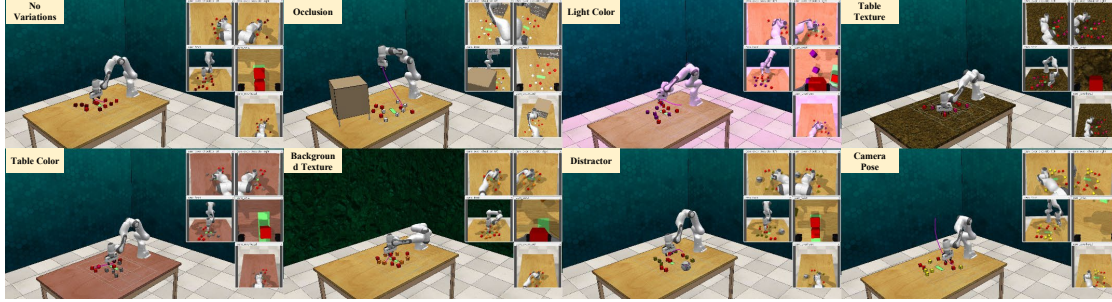
rates across multiple manipulation scenarios. TVVE effectively translates visual completeness into manipulation success, validating that dynamic “seeing” fundamentally underpins robust “acting.”

References

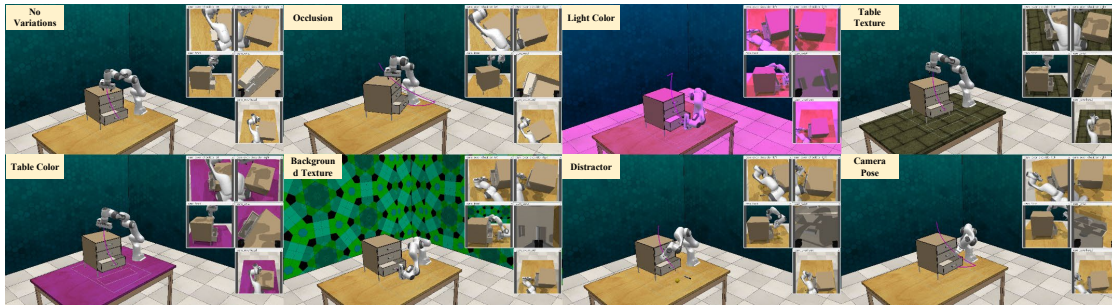
- [1] Shizhe Chen, Ricardo Garcia, Cordelia Schmid, and Ivan Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. In *7th Conference on Robot Learning (CoRL 2023)*, 2023. [5](#)
- [2] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. *IJRR*, 2023. [5](#)
- [3] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *Conference on Robot Learning*, pages 3949–3965. PMLR, 2023. [5](#)
- [4] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. *arXiv preprint arXiv:2306.14896*, 2023. [5](#)
- [5] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024. [4](#), [5](#)
- [6] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia Pinel, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. In *Conference on Robot Learning*, pages 175–187. PMLR, 2023. [5](#)
- [7] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. RL Bench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020. [1](#)
- [8] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022. [5](#)
- [9] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *Conference on Robot Learning*, pages 1949–1974. PMLR, 2025. [5](#)
- [10] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024. [9](#)
- [11] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017. [2](#)
- [12] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023. [5](#)
- [13] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Thomas Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, et al. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. In *Conference on Robot Learning*, pages 649–665. PMLR, 2025. [2](#)
- [14] Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on robot learning*, pages 284–301. PMLR, 2023. [5](#)
- [15] Xinyu Zhang, Yuhan Liu, Haonan Chang, Liam Schramm, and Abdeslam Boularias. Autoregressive action sequence learning for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025. [5](#)



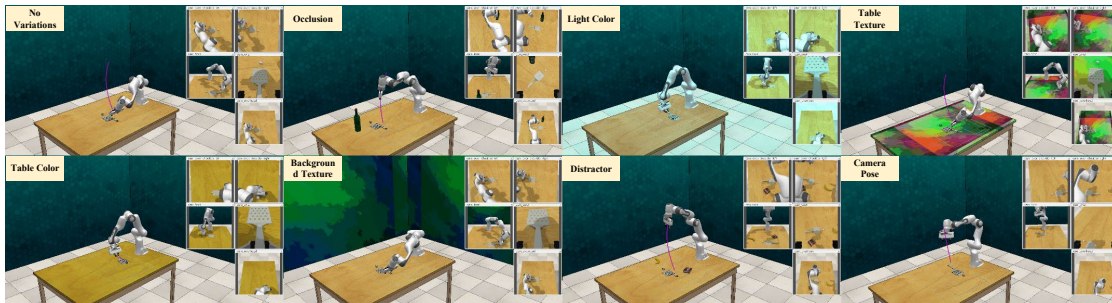
(a) Basketball In Hoop: pick up the basketball and put it in the hoop.



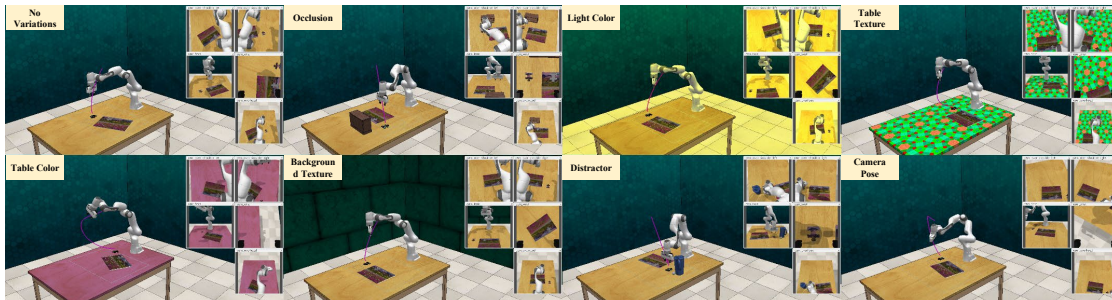
(b) Block Pyramid: stack red blocks in a pyramid.



(c) Close Drawer: close of the bottom drawer by pushing it shut.

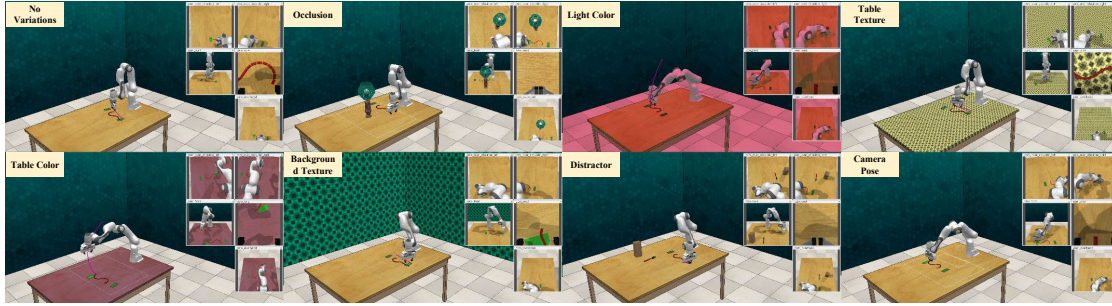


(d) Scoop With Spatula: scoop up the cube and lift it with the spatula.

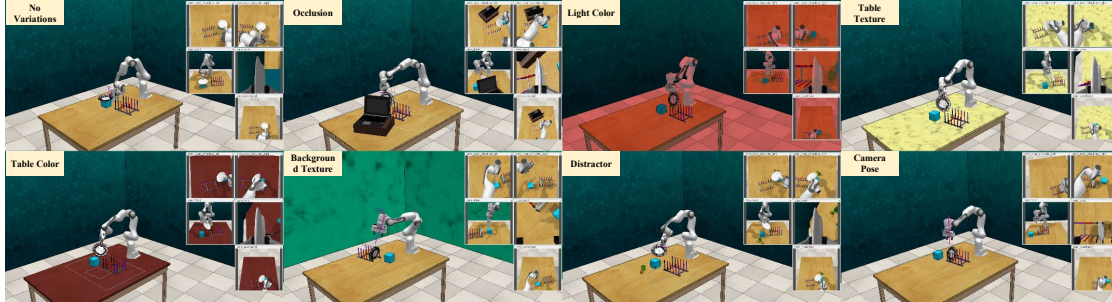


(e) Solve Puzzle: pick up the puzzle piece and place it on the puzzle.

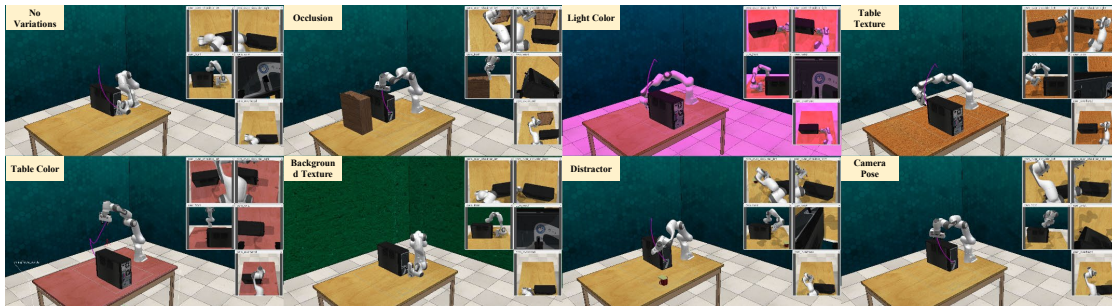
Figure 8. Visualization of different variants for the **basketball_in_hoop**, **block_pyramid**, **close_drawer**, **scoop_with_spatula**, **solve_puzzle** tasks.



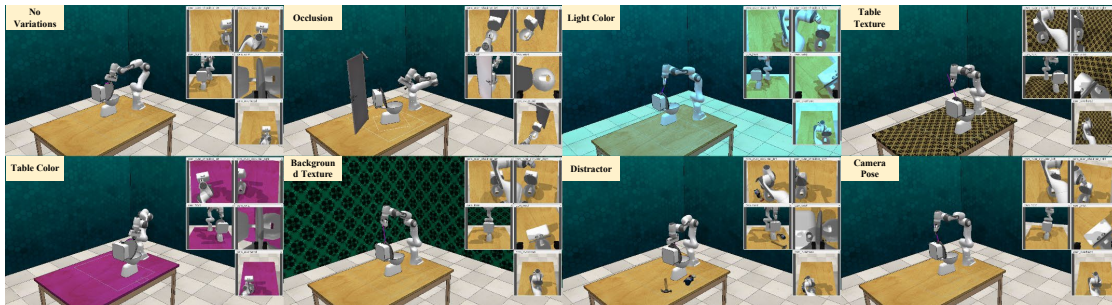
(f) Straighten Rope: grasping each end of the rope in turn, leave the rope straight.



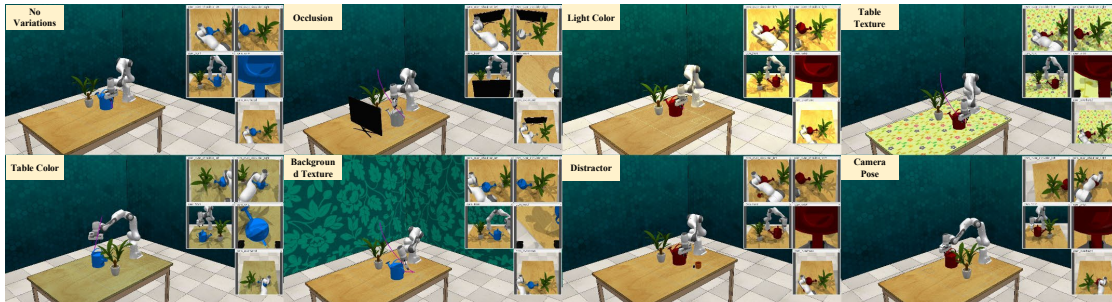
(g) Take Plate Off Colored Dish Rack: remove the dish from the black rack.



(h) Take Usb Out Of Computer: grasp the usb stick and slide it out of the pc.



(i) Toilet Seat Down: grasping the top of the lid, close the toilet seat.



(j) Water Plants: pick up the watering can by its handle and water the plant.

Figure 9. Visualization of different variants for the **straighten_rope**, **take_plate_off_colored_dish_rack**, **take_usb_out_of_computer**, **toilet_seat_down**, **water_plants** tasks.

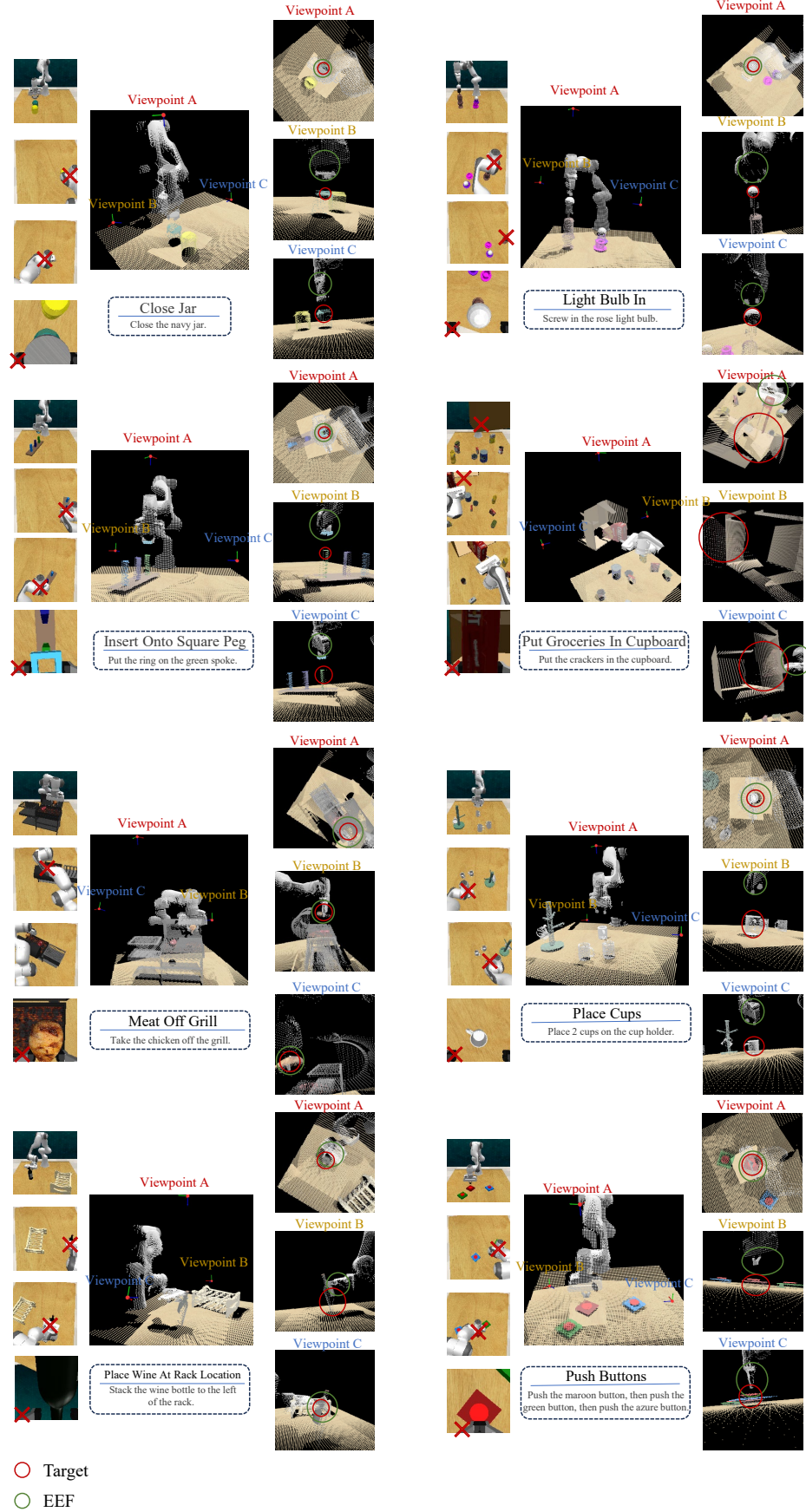


Figure 10. The dynamic multi-view re-rendering visualization results of the **Light Bulb In**, **Close Jar**, **Insert Onto Square Peg**, **Put Groceries In Cupboard**, **Grill**, **Place Cups**, **Place Wine At Rack Location**, and **Push Buttons** tasks.

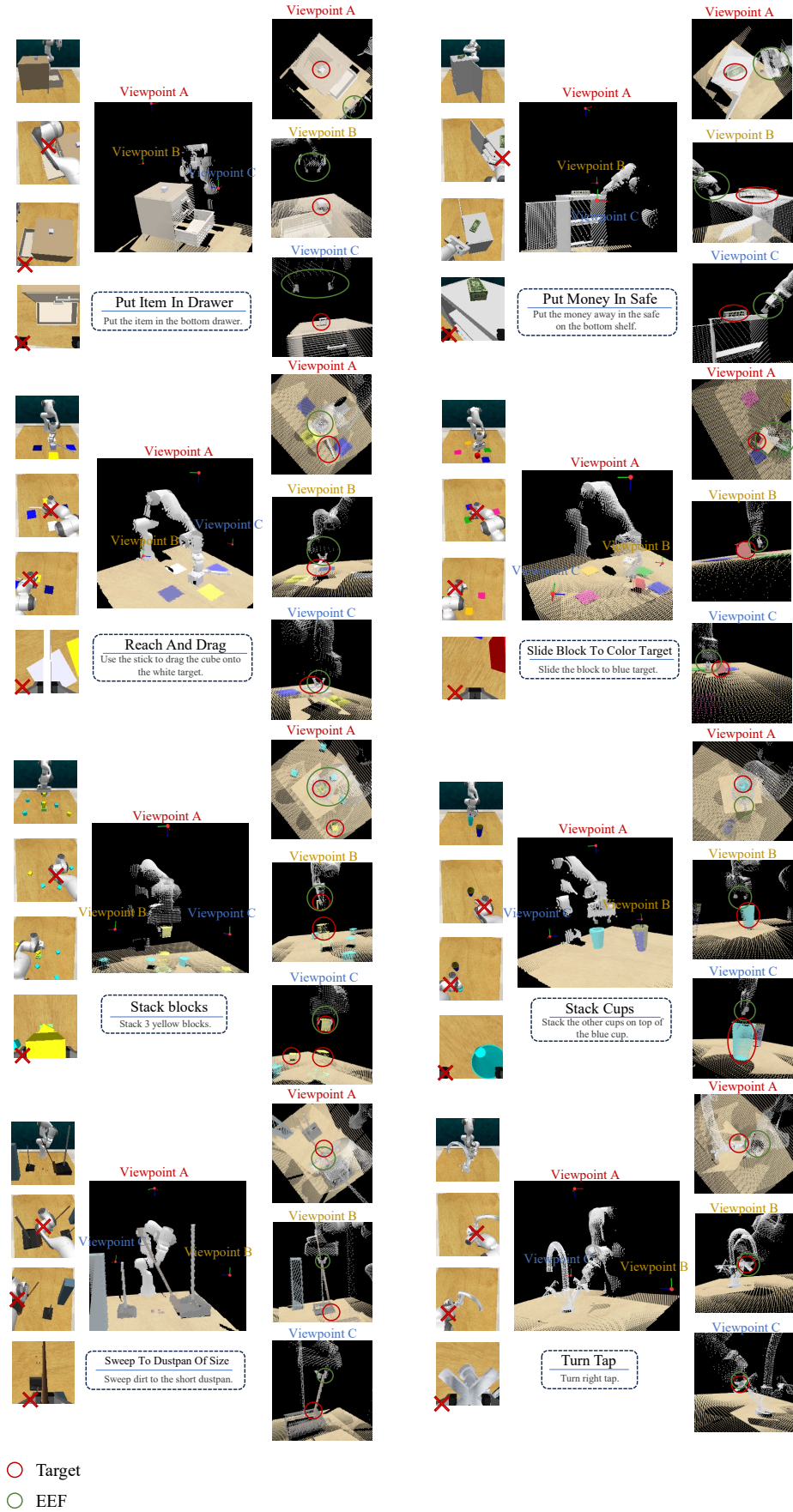


Figure 11. The dynamic multi-view re-rendering visualization results of the **Put Item In Drawer**, **Put Money In Safe**, **Reach And Drag**, **Slide Block To Color Target**, **Stack Blocks**, **Stack Cups**, **Sweep To Dustpan Of Size**, and **Turn Tap** tasks.