LEARNING GEOMETRICALLY-GROUNDED AMODAL 3D REPRESENTATIONS FOR VIEW-GENERALIZABLE ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world robotic manipulation requires visuomotor policies capable of robust 3D scene reasoning under varying camera viewpoints. While recent 3D-aware manipulation policies have shown promise, they still face several limitations: (i) reliance on multi-view observations during inference, which is impractical in camera-constrained deployments; (ii) insufficient geometric fidelity in learned representations, limiting precise and viewpoint-robust control; and (iii) the lack of effective mechanisms for transferring pretrained 3D representations to downstream visuomotor policies. To address these challenges, we present ***GEM3D*** (Geometrically-Grounded 3D Manipulation), a unified 3D representation and policy learning framework for view-generalizable robotic manipulation. *GEM3D* learns amodal 3D representations from single-view RGB-D observations by jointly enforcing point-cloud reconstruction and Gaussian-splatting-based novel-view consistency during pretraining. A multi-step distillation strategy then transfers the learned geometric understanding into a deployable single-view visuomotor policy for downstream manipulation control. Experiments on 12 RLBench tasks and three real-world tasks demonstrate strong manipulation performance. Further zero-shot evaluations on eight RLBench tasks and the real-world tasks show robust generalization to unseen camera poses and substantial gains over existing methods.

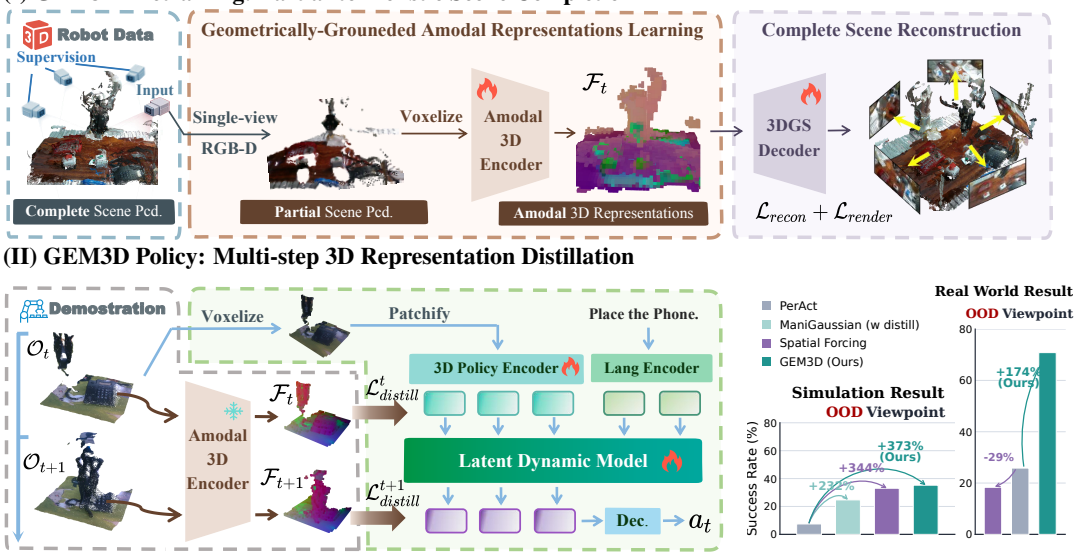


Figure 1. Overview of *GEM3D*. *GEM3D* comprises two key components: (1) *GEM3D Pretraining*, which learns amodal 3D representations via auxiliary scene reconstruction tasks; and (2) *GEM3D Policy*, which distills pretrained 3D representations into a visuomotor policy for view-generalizable manipulation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Amodal (Wu et al., 2025; Ao et al., 2023; Zhou & Tai, 2025) refers to the ability to infer the complete spatial structure of a scene from partial observations, such as a single-view input. In real-world manipulation, robots often act under partial observability caused by occlusions and limited viewpoints. We therefore argue that learning amodal representations is essential for building effective 3D policies.

Learning end-to-end visuomotor policies enables robots to perceive their surroundings and act upon visual understanding (Chi et al., 2023; Kim et al., 2024; Chi et al., 2024; Hansen et al., 2024; Zhang et al., 2024; Ha & Schmidhuber, 2018; Hafner et al., 2021). However, real-world environments are often spatially complex and partially occluded, and camera viewpoints may vary across deployments. Robust manipulation therefore requires policies that can infer amodal 3D scene structure from partial observations and make decisions that generalize well across viewpoints.

Most existing visuomotor policies rely on 2D visual encoders that compress images into latent states (Yarats et al., 2020; Laskin et al., 2020; Hafner et al., 2020). To improve spatial reasoning, recent methods leverage multi-view inputs to learn 3D-aware visual representations (Goyal et al., 2023; Qian et al., 2025; Shridhar et al., 2023). However, their reliance on synchronized multi-view observations makes deployment difficult in practical robotic settings with limited camera availability. To reduce this dependency, some single-view 3D policies incorporate reconstruction objectives, such as Neural Radiance Fields (NeRFs) (Mildenhall et al., 2022) or Gaussian Splatting (Kerbl et al., 2023), to obtain 3D-aware perception (Driess et al., 2022; Wang et al., 2024; Lu et al., 2024; Ze et al., 2023b; 2024). Nevertheless, these methods mainly reconstruct geometry directly observable from the input view and often struggle to infer occluded scene structures.

To address these limitations, we propose ***GEM3D*** (Geometrically-Grounded 3D Manipulation), a two-stage 3D representation-policy learning framework for view-generalizable robotic manipulation. In the first stage, *GEM3D* learns amodal 3D representations from single-view RGB-D inputs by using multi-view observations only as geometric supervision. Given a partial observation from one camera, the model is trained to reconstruct the complete scene point cloud, and enforce view-consistent geometric reasoning through Gaussian-splatting-based novel-view supervision. Such a representation is well suited for manipulation, as it enables the robot to infer the scene structure from any single camera observation, including occluded task-relevant objects and structures.

In the second stage, we transfer the learned spatial understanding to a single-view visuomotor policy through a multi-step distillation paradigm. Directly finetuning a pretrained 3D encoder with action supervision can easily degrade its geometric reasoning capability. To preserve the learned spatial understanding, we freeze the pretrained representation model as a geometric teacher and distill its amodal scene representations into a lightweight policy encoder. We therefore use task execution from unseen camera viewpoints as a practical measure of whether pretrained 3D knowledge is preserved in the policy. As a result, even when trained only on fixed single-view demonstrations, *GEM3D* can perform manipulation from out-of-distribution camera viewpoints at inference time.

We evaluate *GEM3D* on three real-world tasks and 12 RLBench tasks, achieving 44.2% average in-domain success on the latter, 9.1 points above the strongest baseline. We further test view generalization on eight RLBench tasks, reaching 41.0%/29.5% under moderate/large shifts, 11.7/9.4 points above the strongest baselines. In the real world, *GEM3D* reaches 88.3% at the training view and 78.3%/63.3% under moderate/large shifts, exceeding the strongest baselines by 43.3, 41.7, and 48.3 points, respectively.

In summary, the contributions of *GEM3D* are threefold: (i) We propose a single-view amodal 3D representation learning framework for view-generalizable robotic manipulation that infers occlusion-aware scene geometry from partial RGB-D observations. (ii) We introduce a geometry-preserving distillation strategy that transfers pretrained spatial reasoning capability into a lightweight deployable visuomotor policy. (iii) We demonstrate strong manipulation performance on 12 RLBench tasks, with further viewpoint-generalization evaluations on eight RLBench tasks and three real-world tasks.

2 Related Work

3D Representation Learning. Recent visuomotor policy learning has explored diverse visual representations, from 2D image features (Chi et al., 2023; Zhang et al., 2026) to explicit 3D structures such as voxel grids (Shridhar et al., 2023) and point clouds (Chen et al., 2023; Guhur et al., 2023; Ze et al., 2024; Chen et al., 2026). Among these, novel-view synthesis (NVS) has emerged as a useful auxiliary objective for learning 3D-aware representations, as rendering unseen viewpoints encourages the model to infer scene geometry beyond the input observation (Mildenhall et al., 2022; Kerbl et al., 2023; Poole et al., 2022; Liu et al., 2023; Shi et al., 2023). However, accurate and generalizable reconstruction remains challenging: classical NeRFs (Mildenhall et al., 2022) and Gaussian Splatting (Kerbl et al., 2023) often degrade outside the training distribution, and feed-forward Gaussian Splatting methods (Szymanowicz et al., 2024; Ren et al., 2024; Charatan et al., 2024), while more generalizable, still exhibit degraded geometry and texture in dynamic robotic scenes. These limitations highlight the need for an NVS framework that achieves both high-fidelity reconstruction and robust generalization across diverse robotic scenes.

Visuomotor Policy Representation Learning. Auxiliary representation learning for control mainly follows pretrain-finetune or joint-training paradigms. Pretrain-finetune methods initialize visual encoders from large-scale 2D or 3D perception datasets before adapting them to downstream control (Xiao et al., 2022; Wang et al., 2022; Yan et al., 2024; Ze et al., 2023a; Qian et al., 2025; Wang et al., 2026b), but domain discrepancies between perception pretraining and policy learning can cause suboptimal transfer or even degrade control performance (Hansen et al., 2022; Hou et al., 2025). Joint-training methods optimize visuomotor policies together with auxiliary objectives such as NVS (Ze et al., 2023b; Lu et al., 2024), yet the competing objectives can lead to imbalanced optimization, where the learned representation remains weak and the policy still overfits to training viewpoints. In contrast to prior works, we propose a multi-step distillation framework that softly guides policy encoder training with pretrained 3D representations, effectively preserving geometric knowledge acquired during pretraining.

3 Method

3.1 Problem Formulation

We consider language-conditioned robotic manipulation from a single RGB-D camera. At each timestep t, the robot receives $o_t^v = \{I_t^v, D_t^v, K_t^v\}$, where v is the camera viewpoint and I_t^v, D_t^v, and K_t^v are the RGB image, depth map, and camera parameters. Given a language instruction l, our goal is to learn a policy $\pi(a_t|o_t^v, l)$ that predicts action a_t from an arbitrary-viewpoint observation.

Pretraining phase. GEM3D learns a 3D representation F_t from a multi-view dataset. Given a single-view input o_t^v, the remaining views $v' \neq v$ at the same timestep provide cross-view supervision. The goal is to make F_t infer amodal 3D information from the partial observation.

Policy learning phase. We then train a 3D visuomotor policy on single-view datasets with a separate policy encoder. This encoder maps o_t^v into a policy representation x_t, which is used to predict the action a_t. The goal is to transfer the 3D understanding pretrained in F_t into x_t, enabling geometry-aware action prediction from single-view inputs.

Inference. We evaluate zero-shot view generalization as a practical proxy for 3D scene understanding by testing each episode from a random camera viewpoint unseen during policy training.

3.2 Geometrically-Grounded Pretraining

GEM3D's pretraining pipeline consists of three main components: (i) a ***3D Feature Extraction*** module that encodes single-view RGB-D observations into *Amodal 3D Representations*; (ii) a ***Point Cloud Reconstruction*** objective that learns amodal 3D scene geometry; and (iii) a ***Feed-forward 3D Gaussian Splatting*** module that captures fine-grained texture through multi-view supervision. The detailed pipeline is illustrated in Fig. 2.

3D Feature Extraction. As shown in Fig. 1, we represent the scene with a voxel-based 3D feature $F_t \in \mathbb{R}^{D^3 \times C}$ encoded from a single-view RGB-D observation $o_t^v = \{I_t^v, D_t^v, K_t^v\}$. Voxels

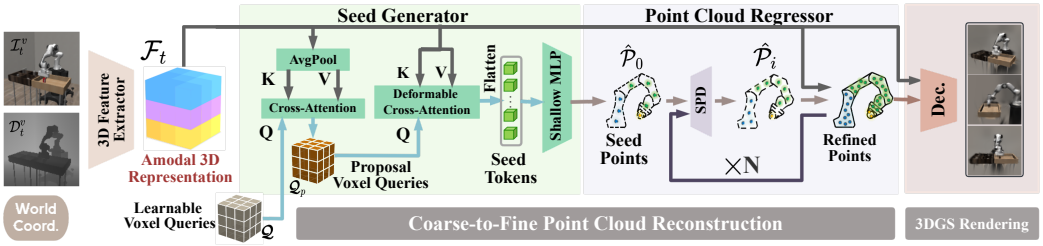


Figure 2. ***GEM3D* 3D Pretraining Pipeline.** Given single-view RGB-D volumetric features, we (i) first capture the coarse scene layout and local geometric details via two-stage attention, (ii) use Snowflake Point Deconvolution (SPD) blocks to progressively double the point cloud, and (iii) apply Gaussian-splatting-based novel-view rendering for fine-grained texture supervision.

provide a structured 3D representation in which each token corresponds to a local spatial region, making them well suited for preserving geometric layout. While the input RGB-D view only covers part of the scene, GEM3D pretraining encourages F_t to capture amodal spatial information across viewpoints; we therefore refer to F_t as an *Amodal 3D Representation*. The observation is lifted into 3D, voxelized into occupancy and semantic feature volumes, and encoded into F_t by a 3D U-Net encoder (Çiçek et al., 2016); further implementation details are provided in Appendix A.2.

Coarse-to-Fine Point Cloud Reconstruction To supervise amodal scene geometry, we sample all remaining viewpoints $v' \neq v$, back-project their depth maps into the world coordinate frame, and merge the resulting points into a complete point cloud P_{full}.

As shown in Fig. 2 Given F_t, GEM3D predicts the multi-view-complete point cloud in a coarse-to-fine manner, as shown in Fig. 2. Learnable voxel queries $Q \in \mathbb{R}^{d^3 \times C}$ first extract seed tokens from F_t through a two-stage attention design: standard cross-attention captures the coarse scene layout from a downsampled feature, while 3D deformable cross-attention refines the tokens with local geometric details from the full-resolution feature. The refined seed tokens are decoded into seed points $\hat{P}_0 \in \mathbb{R}^{d^3 \times 3}$, which are then progressively densified by Snowflake Point Deconvolution (SPD) blocks (Xiang et al., 2021). At each refinement stage, each point generates r child points, expanding $\hat{P}_{i-1} \in \mathbb{R}^{r^{i-1} d^3 \times 3}$ to $\hat{P}_i \in \mathbb{R}^{r^i d^3 \times 3}$ and yielding multi-scale predictions $\{\hat{P}_i\}$ for reconstruction supervision. Implementation details are provided in Appendix A.3.

At each refinement stage, we randomly sample a subset $P_i \subset P_{\text{full}}$ with the same size as $\hat{P}_i$ and supervise the prediction with a symmetric Chamfer distance:

$$L_{\text{rec}} = \sum_i \left(\frac{1}{|\hat{P}_i|} \sum_{\hat{p} \in \hat{P}_i} \min_{p \in P_i} \|\hat{p} - p\|_2 + \frac{1}{|P_i|} \sum_{p \in P_i} \min_{\hat{p} \in \hat{P}_i} \|p - \hat{p}\|_2 \right). \tag{1}$$

Feed-forward 3D Gaussian Splatting. The reconstructed point cloud $\hat{\mathcal{P}} \in \mathbb{R}^{N_p \times 3}$ provides explicit geometric anchors, while manipulation also requires fine-grained appearance cues such as object texture and boundaries. We use $\hat{\mathcal{P}}$ as Gaussian centers μ, sample features $\mathcal{F}_{\mathcal{G}} \in \mathbb{R}^{N_p \times C}$ from $\mathcal{F}_t$ at these locations, and regress the remaining Gaussian parameters with a ResNetFC network:

$$\{c, \sigma, r, s\} = \text{ResNetFC}(\mathcal{F}_{\mathcal{G}}), \tag{2}$$

where c, σ, r, and s denote color, opacity, rotation, and scale, respectively. The resulting Gaussians $\mathcal{G} = \{\mu, c, \sigma, r, s\}$ are rendered from held-out viewpoints $v' \neq v$ with a differentiable splatting renderer and supervised by the corresponding RGB observations $\mathcal{I}^t_{v'}$. We use a focal loss (Lin et al., 2017), which assigns higher weights to hard-to-predict pixels and therefore emphasizes dynamic or visually ambiguous regions:

$$\mathcal{L}_{\text{rend}} = \text{FocalLoss}(\hat{\mathcal{I}}^t_{v'}, \mathcal{I}^t_{v'}). \tag{3}$$

The overall pretraining objective combines reconstruction and rendering supervision:

$$\mathcal{L}_{pretrain} = \mathcal{L}_{\text{rec}} + \lambda_{\text{rend}} \mathcal{L}_{\text{rend}}. \tag{4}$$

where λ_{rend} balances the two objectives.



Figure 3. Real-world evaluation settings. The upper right shows the three evaluated tasks. We use two-stage scoring, with each stage marked by I and II in the figure. For example, in the *Place Phone* task, correctly picking up and placing down the receiver are counted as two separate success rates. The Stage II success rate also corresponds to the full-task success rate. During evaluation, we test generalization to out-of-distribution (OOD) camera viewpoints, tabletop textures, and object colors.

3.3 Distillation-based Policy Learning

Instead of directly finetuning the pretrained 3D encoder, which may cause representation drift, we freeze it as a geometric teacher and train a separate PerceiverIO-style policy encoder (Jaegle et al., 2021). Before policy encoding, each single-view observation o_t^v is calibrated to the same world coordinate frame as pretraining. Then the policy encoder produces latent tokens $\mathbf{x}_t$ from o_t^v, while the frozen teacher output $\mathcal{F}_t$ is patchified into reference tokens $\tilde{\mathbf{x}}_t$. We align $\mathbf{x}_t$ with $\tilde{\mathbf{x}}_t$ using a cosine-similarity distillation loss to transfer the pretrained 3D knowledge to the policy.

For robotic control, each action can be interpreted as inducing a transition from the current latent state to the next latent state. We therefore factorize the policy as $p(\mathbf{x}_{t+1}|\mathbf{x}_t, l)\, p(a_t|\mathbf{x}_{t+1})$: a latent dynamics model first predicts $\mathbf{x}_{t+1}$ from $\mathbf{x}_t$ together with proprioceptive and language embeddings, and an action head then decodes a_t from $\mathbf{x}_{t+1}$, as illustrated in Fig. 1. This design enables multi-step distillation, which injects pretrained 3D knowledge and regularizes temporal transition modeling by aligning the current and predicted next latent states with their corresponding teacher tokens:

$$\mathcal{L}_{\text{multi-distill}} = \mathcal{L}_{\text{distill}}^{t} + \mathcal{L}_{\text{distill}}^{t+1} = \mathcal{L}_{\cos}(\mathbf{x}_t, \text{sg}[\tilde{\mathbf{x}}_t]) + \mathcal{L}_{\cos}(\mathbf{x}_{t+1}, \text{sg}[\tilde{\mathbf{x}}_{t+1}]). \tag{5}$$

Note that policy learning uses only single-view datasets, and **neither training nor inference requires explicit geometric reconstruction**. We train the policy with imitation and distillation losses:

$$\mathcal{L}_{\text{policy}} = \mathcal{L}_{\text{action}} + \lambda_{\text{distill}} \mathcal{L}_{\text{multi-distill}}. \tag{6}$$

where $\mathcal{L}_{\text{action}}$ supervises action prediction and λ_{distill} balances the two objectives.

Details of the module design and loss hyperparameter settings are provided in Appendix A.1.

4 Experiments

4.1 Experimental Setup

Evaluation setup. For real-world experiments, we pretrain *GEM3D* on 48 RH20T (Fang et al., 2023) tasks, with 7–9 viewpoints per task. For policy learning, we choose three unseen manipulation tasks and collect 50 single-view demonstrations for each task on our Flexiv setup: (i) *Place Phone*, placing a receiver onto the phone base; (ii) *Stack Cups*, stacking one cup onto another; and (iii) *Retrieve from Drawer*, retrieving an object from a drawer and closing it. Since the baseline methods have low full-task success rates in the real world, each task is scored with two execution stages, as detailed in Fig. 3.

In simulation, we evaluate in-domain performance on 12 RLBench (James et al., 2020) tasks across nine scenes with diverse language instructions. During pretraining, multi-view observations are

Table 1. Simulation experimental results. Comparison of *GEM3D* with state-of-the-art methods using the training view for in-domain evaluation (SR%). Avg. SR is computed over all 12 RLBench tasks, and the remaining columns show six representative tasks. **Bold** numbers denote the best performance, and underlined numbers denote the second best.

Methods	12-task Avg. SR	Take Money out Safe	Close Jar	Meat on Grill	Water Plants	Turn Tap	Put Money in Safe
PerAct	21.9 ± 5.3	13.7 ± 2.9	19.0 ± 4.6	16.0 ± 7.9	8.3 ± 2.3	43.7 ± 4.7	10.7 ± 0.6
GNFactor	29.2 ± 18.4	19.0 ± 4.0	32.0 ± 16.8	35.0 ± 6.1	22.0 ± 4.4	40.7 ± 10.7	11.0 ± 10.5
ManiGaussian	31.5 ± 8.5	49.3 ± 10.5	27.3 ± 16.4	30.7 ± 12.9	19.3 ± 9.0	44.0 ± 10.5	5.7 ± 3.5
DP3	33.0 ± 7.6	31.6 ± 7.1	29.7 ± 8.6	55.5 ± 8.4	21.6 ± 6.4	46.8 ± 8.7	15.4 ± 4.6
ManiGaussian (w Distill)	33.3 ± 10.4	43.0 ± 23.4	31.0 ± 12.1	48.7 ± 2.9	16.3 ± 8.7	42.7 ± 14.6	20.3 ± 9.2
Spatial Forcing	35.1 ± 7.9	46.9 ± 10.1	31.4 ± 9.4	47.1 ± 8.1	21.3 ± 6.3	46.5 ± 9.0	24.1 ± 7.3
***GEM3D** (Ours)*	**44.2** ± 6.2	**72.0** ± 13.7	**44.7** ± 3.2	**61.0** ± 22.1	**32.7** ± 3.5	**54.0** ± 7.2	**29.7** ± 2.9

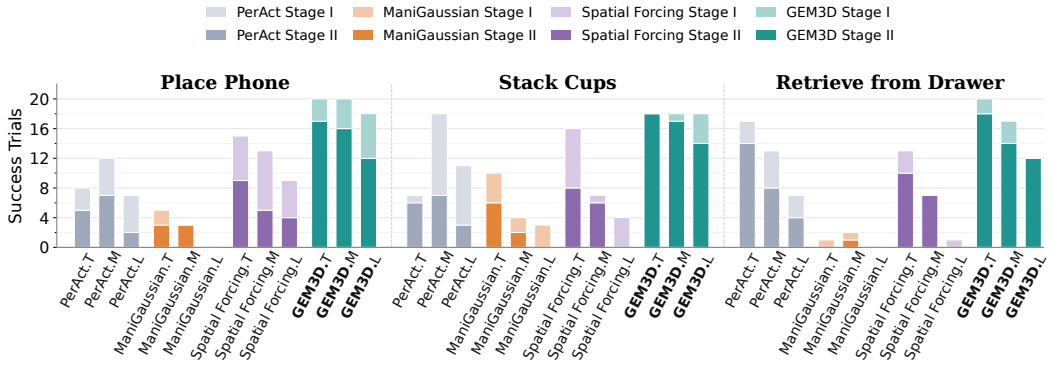


Figure 4. Real-world experimental results. We report the number of successful trials out of 20 evaluations of different stages. For each task, we evaluate three viewpoint settings: *T* denotes the training view, *M* denotes a moderate viewpoint shift, and *L* denotes a large viewpoint shift.

captured from eight uniformly distributed surrounding cameras. For policy learning, each task is trained using 20 expert demonstrations collected from a fixed front-view camera. In addition, we conduct a Robosuite (Zhu et al., 2020) reinforcement learning experiment to evaluate the generality of our amodal 3D representation beyond imitation learning; details are provided in Appendix (D.7).

Evaluation metrics. To assess view generalization in both simulation and real-world settings, we conduct zero-shot evaluations under three camera configurations, as shown in Fig. 3: *Training View*, where the camera viewpoint is the same as policy training; *Moderate Shift*, where the camera viewpoint is shifted within 30° around the robot base; and *Large Shift*, where the camera viewpoint is shifted between 60° and 90°. For all methods, RGB-D observations from different viewpoints are calibrated into the same world coordinate frame during evaluation. However, different viewpoint shifts induce different levels of incompleteness in the single-view point cloud, allowing us to assess each policy's implicit amodal scene inference ability.

For real-world experiments, each task-viewpoint setting is evaluated with 20 trials, for a total of $3 \times 3 \times 20 = 180$ trials for each method. For simulation, all models are trained with three fixed random seeds (0, 1, 2), and each seed is evaluated on 100 test episodes. In both real-world and simulation evaluations, we randomize tabletop textures and object colors that are unseen during policy training. On RLBench, *GEM3D* runs at 1.28 Hz, compared with 1.18 Hz for *PerAct* and 1.36 Hz for *ManiGaussian* (Appendix D.2).

Baselines. We first include **3D Diffusion Policy (DP3)** (Ze et al., 2024) as a general-purpose 3D policy baseline. To accommodate *GEM3D*'s voxel representations, we adopt **PerceiverIO** (Jaegle et al., 2021) as the policy backbone. We use **PerAct** (Shridhar et al., 2023) as a basic naive baseline without auxiliary co-training, distillation, or multi-view pretraining. All the following methods, including our *GEM3D* , use the same PerceiverIO architecture. Other baselines include **GNFactor** (Ze et al., 2023b), which uses NeRF-based auxiliary representation learning, and **ManiGaussian** (Lu et al., 2024), which jointly trains a 3D Gaussian Splatting representation with the policy. We further combine alternative representations with our distillation strategy. Specifically, **ManiGaussian (w/ Distill)** distills the pretrained ManiGaussian representations into the PerceiverIO backbone. We

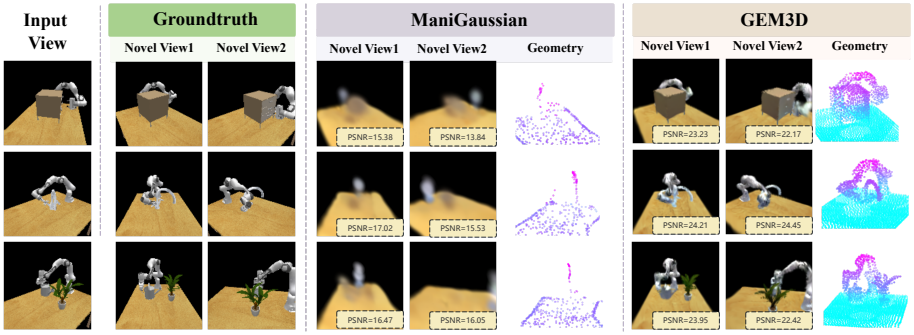


Figure 5. Reconstruction quality in simulation. Given a single-view input, we compare the novel-view rendering and point cloud reconstruction results of *ManiGaussian* and *GEM3D* .

also adapt **Spatial Forcing** (Li et al., 2026) by unprojecting a frozen **VGGT-Ω** (Wang et al., 2026a) representation into calibrated 3D voxels and distilling it into the same PerceiverIO backbone. These two baselines serve as contrasts to our amodal GEM3D representation, allowing us to isolate the benefits of our representation from those of the distillation strategy.

4.2 Experimental Results

4.2.1 Comparison with State-of-the-Art Methods

In-domain evaluation. In real-world experiments, Fig. 4 shows that *GEM3D* achieves an average Stage II full-task success rate of 88.3% across the three tasks, outperforming *Spatial Forcing* (45.0%) by 43.3 points and *PerAct* (41.7%) by 46.6 points. In simulation, Table 1 reports six representative RLBench tasks and the 12-task average, where *GEM3D* obtains an average success rate of 44.2%, improving over the strongest baseline, *Spatial Forcing* (35.1%), by 9.1 points. *ManiGaussian (w Distill)* improves the original *ManiGaussian* from 31.5% to 33.3% but remains 10.9 points below *GEM3D* , showing that both distillation and geometry-grounded representations matter for manipulation.

The weak real-world performance of *ManiGaussian* in Fig. 4 suggests that joint reconstruction and control learning is unstable under noisy, incomplete physical observations. Its rendering objective can compete with action supervision, and the degraded reconstructions in Fig. 5 and Fig. D.5 support this explanation.

Generalization evaluation. We further evaluate view generalization on eight RLBench tasks (Table D.6): *PerAct* drops from 29.7% at the training view to 9.9%/5.0% under moderate/large shifts, while *DP3* drops from 42.2% to 18.6%/11.5%. Although 3D policies are commonly expected to be visually robust, a viewpoint change reshapes the partially observed point cloud even after world-frame calibration, causing severe policy degradation under larger shifts. In comparison, *GEM3D* reaches 41.0%/29.5%, exceeding the strongest baselines by 11.7/9.4 points.

Since real-world evaluation includes unseen tabletop textures and object colors, the high training-view success also reflects appearance robustness inherited from diverse RH20T pretraining. Under shifted cameras, *GEM3D* achieves 78.3% Stage II success for moderate shifts and 63.3% for large shifts, compared with 36.7%/15.0% for *PerAct* and 30.0%/6.7% for *Spatial Forcing*. Our adapted *Spatial Forcing* adds VGGT-Ω distillation to the PerAct backbone and improves training-view success from 41.7% to 45.0%, yet lowers OOD success below PerAct. We hypothesize that VGGT-Ω reconstructs only content visible from the current camera, so distilling its view-specific incomplete geometry biases the policy toward the training-view observation pattern. In contrast, *GEM3D* targets amodal completion and encourages consistent full-scene structure inference, yielding representations less sensitive to input appearance and viewpoint changes. On the first two tasks, its Stage II success across the three views has standard deviations of only 10.8 points for *Place Phone* and 8.5 points for *Stack Cups*.

Table 2. Ablations on *GEM3D Pretraining* and *GEM3D Policy* . We select three representative RLBench tasks and report the average success rate under different viewpoint shifts (SR%).

Num. Views	Training	Moderate	Large
8	58.8 ± 6.0	45.9 ± 8.4	39.6 ± 8.5
4	54.6 ± 4.9	42.4 ± 6.6	37.3 ± 8.8
2	54.9 ± 5.1	40.3 ± 5.1	33.6 ± 6.8
1	47.2 ± 7.4	33.4 ± 6.1	30.8 ± 5.8

(a) Ablation on the number of pretraining viewpoints.

Methods	Training	Moderate	Large
Pretrain-Finetune	32.2 ± 11.1	21.1 ± 6.7	15.1 ± 6.8
Single-step Distillation	54.6 ± 7.6	41.7 ± 9.0	37.7 ± 8.7
GEM3D *(ours)*	58.8 ± 6.0	45.9 ± 8.4	39.6 ± 8.5

(b) Ablation on different policy training paradigms.

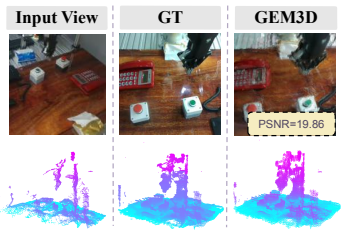


Figure 6. Image and point cloud reconstruction quality in the real world.

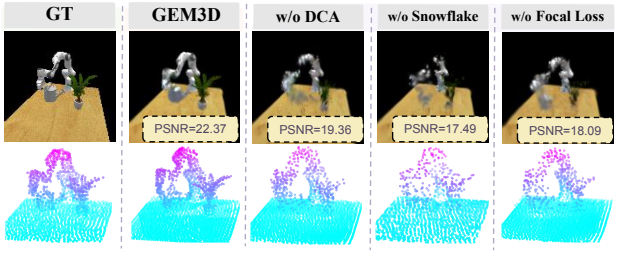


Figure 7. Qualitative ablation of *GEM3D Pretraining* designs. Novel-view rendering and point cloud reconstruction results for ablated variants.

Reconstruction Analysis. Fig. 5 and Fig. 6 show that *GEM3D Pretraining* recovers clearer boundaries and denser geometry than *ManiGaussian*; its 22.17–24.45 dB PSNR also substantially exceeds 13.84–17.02 dB, supporting stronger scene reconstruction for manipulation.

4.3 Ablations

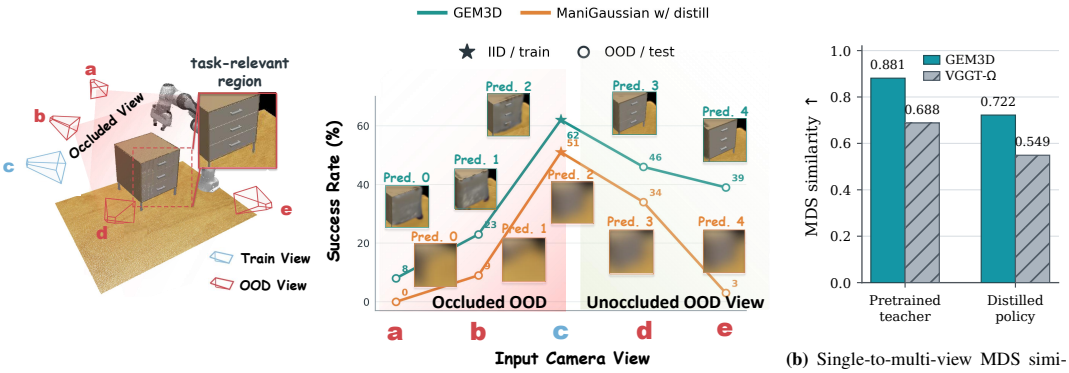
Ablation on GEM3D Pretraining We first ablate the architectural designs in *GEM3D Pretraining* through three variants in Fig. 7: ***w/o DCA***, which replaces deformable cross-attention with standard cross-attention between dense and downsampled volumetric features; ***w/o Snowflake***, which replaces coarse-to-fine point cloud reconstruction with single-step reconstruction; and ***w/o Focal Loss***, which replaces focal loss with MSE for Gaussian rendering. Removing any component clearly degrades reconstruction quality, causing collapsed robot arms, missing plant structures, and less consistent textures.

We further study the dependence on multi-view supervision by varying the number of pretraining views and conducting multi-task training on `Phone on Base`, `Meat on Grill`, and `Turn Tap`. As shown in Table 2a, reducing the number of views from eight to two causes only marginal performance drops across viewpoint conditions, suggesting that *GEM3D Pretraining* remains effective under limited multi-view supervision. We also include a one-view setting, where Gaussian Splatting rendering is disabled because no held-out view is available for novel-view supervision. This setting leads to a nearly 10% drop, showing that Gaussian Splatting is critical for capturing fine-grained appearance details and enforcing cross-view spatial consistency.

Ablation on GEM3D Policy We ablate the *GEM3D Policy* architecture by comparing two variants: *Pretrain-Finetune*, which directly fine-tunes the pretrained encoder without distillation, and *Single-step Distillation*, which removes the $t{+}1$ latent distillation step and aligns only input-level tokens.

Table 2b reports the average success rates across three tasks. The results indicate that incorporating distillation at the $t{+}1$ step improves temporal consistency in the latent dynamics, leading to noticeably stronger policy performance. In contrast, conventional pretrain-finetune suffers from representation drift due to the domain gap between pretraining and downstream tasks (Hansen et al., 2022; Pang et al., 2024), substantially reducing manipulation success.

5 DISCUSSION



(a) Rendering and policy performance.

(b) Single-to-multi-view MDS similarity.

Figure 8. Rendering, policy performance, and cross-view feature alignment. (a) *GEM3D* and *ManiGaussian* reconstructions and seed-0 success over 100 *Open Drawer* episodes per view; policies are trained only at view **c**. **(b)** Single-to-multi-view MDS similarity on paired *Open Drawer* scene states before and after policy distillation. Higher is better.

Q: Do *GEM3D*'s view-generalization gains arise from scene memorization or prior exposure to multiple OOD viewpoints during pretraining? No. Pretraining uses 48 RH20T tasks, whereas deployment involves three unseen tasks in our setup (Fig. 4), with substantially different layouts, equipment, and lighting. These scenes and their OOD observations are absent from pretraining, ruling out direct memorization. Our key idea is to learn amodal representations of complete scenes through large-scale reconstruction from partial single-view inputs under multi-view supervision. Distillation transfers spatial inference and cross-view consistency capabilities to policies trained at only one fixed view, enabling task success, even in novel scenes and from unseen viewpoints.

Q: Does improved rendering quality directly imply higher policy success? Reconstruction quality in task-critical regions is strongly, though not strictly, related to control performance. All policies in Fig. 8(a) are trained only from view **c** and tested without adaptation at OOD views **a**, **b**, **d**, and **e**. Occlusion in rear views **a** and **b** degrades both methods, confirming that missing task-critical evidence remains limiting. Under unoccluded views **d** and **e**, however, *GEM3D* preserves the "drawer" structure and achieves 46% and 39% success, whereas *ManiGaussian* loses this structure and falls to 34% and 3%. This contrast indicates that *GEM3D* captures view-consistent task geometry, while its remaining drop under the largest shifts also argues against a view-agnostic memorized trajectory: control still depends on visually grounded state estimation.

Q: Does *GEM3D* learn more view-consistent representations than VGGT-Ω? For each paired *Open Drawer* state, view-**c** provides one partial point cloud, whereas concatenated calibrated views provide a more complete geometric reference to the same 3D encoder. Alignment therefore tests whether a partial observation preserves scene-state structure beyond view-specific appearance. We quantify it with joint classical MDS (Appendix D.8). In Fig. 8(b), *GEM3D* exceeds VGGT-Ω before distillation (0.881 vs. 0.688) and afterward (0.722 vs. 0.549), showing that its amodal geometry is more view-consistent and better retained by the policy.

6 CONCLUSION & LIMITATIONS

We introduce *GEM3D*, which combines geometrically-grounded amodal 3D pretraining with multi-step policy distillation to improve in-domain manipulation and zero-shot view generalization in simulation and real-world experiments.

GEM3D has two main limitations: pretraining relies on potentially costly multi-view RGB-D supervision, and both stages require observations calibrated to a shared world frame, reducing scalability and introducing sensitivity to calibration errors and setup overhead.

LLM USAGE STATEMENT

We used large language models (LLMs) to assist with language polishing, grammar correction, and improvements to the clarity, readability, and flow of the manuscript. The authors reviewed and revised the LLM-assisted text and take full responsibility for the content and accuracy of the final manuscript.

REFERENCES

Jiayang Ao, Qiuhong Ke, and Krista A Ehinger. Image amodal completion: A survey. *Computer Vision and Image Understanding*, 229:103661, 2023.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19457–19467, 2024.

Ruoqu Chen, Feixiang Ruan, Liu Cao, Zihao Wang, Botian Xu, Shiqin Tong, Jiajun Liu, Mingzhi Pei, Chenyu Zhang, Wanli Xing, Kaifeng Zhang, and Mengdi Xu. Dex-X: Learning visual-tactile dexterous manipulation from human videos with simulated interaction, 2026. URL https://arxiv.org/abs/2609.07747.

Shizhe Chen, Ricardo Garcia, Cordelia Schmid, and Ivan Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. *arXiv preprint arXiv:2309.15596*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots, March 2024.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pp. 424–432. Springer, 2016.

Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16931–16945, 2022.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot, September 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation, June 2023.

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*, pp. 175–187. PMLR, 2023.

David Ha and Jürgen Schmidhuber. World models, March 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. *arXiv preprint arXiv:2212.05749*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, March 2024.

Chengkai Hou, Yanjie Ze, Yankai Fu, Zeyu Gao, Songbo Hu, Yue Yu, Shanghang Zhang, and Huazhe Xu. 4d visual pre-training for robot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8451–8461, 2025.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, August 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, September 2024.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5639–5650. PMLR, November 2020.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=euMVC1DO4k.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9298–9309, 2023.

Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation, July 2024.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, January 2022. ISSN 0001-0782, 1557-7317. doi: 10.1145/3503250.

Yatian Pang, Eng Hock Francis Tay, Li Yuan, and Zhenghua Chen. Masked autoencoders for 3d point cloud self-supervised learning. *World Scientific Annual Review of Artificial Intelligence*, 1: 2440001, 2024. doi: 10.1142/S2811032324400010.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Shengyi Qian, Kaichun Mo, Valts Blukis, David F. Fouhey, Dieter Fox, and Ankit Goyal. 3d-mvp: 3d multiview pretraining for manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22530–22539, June 2025.

Xuanchi Ren, Yifan Lu, Hanxue Liang, Zhangjie Wu, Huan Ling, Mike Chen, Sanja Fidler, Francis Williams, and Jiahui Huang. Scube: Instant large-scale scene reconstruction using voxsplats. *Advances in Neural Information Processing Systems*, 37:97670–97698, 2024.

Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10208–10217, 2024.

Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 32974–32988, 2022.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-ω. *arXiv preprint arXiv:2605.15195*, 2026a.

Jiaxu Wang, Qiang Zhang, Jingkai Sun, Jiahang Cao, Gang Han, Wen Zhao, Weining Zhang, Yecheng Shao, Yijie Guo, and Renjing Xu. Reinforcement learning with generalizable gaussian splatting, August 2024.

Yidi Wang, Feixiang Ruan, Ruoqu Chen, Jie Yin, Yang Yu, Mengdi Xu, and Kaifeng Zhang. R2S-Eval: Robot evaluation with real-to-sim calibration via vision-language models, 2026b. URL `https://arxiv.org/abs/2609.03276`.

Tianhao Wu, Chuanxia Zheng, Frank Guan, Andrea Vedaldi, and Tat-Jen Cham. Amodal3r: Amodal 3d reconstruction from occluded 2d images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9181–9193, 2025.

Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5499–5509, 2021.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

Ge Yan, Yueh-Hua Wu, and Xiaolong Wang. Dnact: Diffusion guided multi-task 3d policy learning. *arXiv preprint arXiv:2403.04115*, 2024.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images, July 2020.

Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 8(5): 2890–2897, 2023a.

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 *Proceedings of The 7th Conference on Robot Learning*, pp. 284–301. PMLR, December 2023b.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations, June 2024.

Di Zhang, Bowen Lv, Hai Zhang, Feifan Yang, Junqiao Zhao, Hang Yu, Chang Huang, Hongtu Zhou, Chen Ye, and Changjun Jiang. Focus on what matters: Separated models for visual-based rl generalization, September 2024.

Hai Zhang, Siqi Liang, Li Chen, Yuxian Li, Yukuan Xu, Yichao Zhong, Fu Zhang, and Hongyang Li. Sparse video generation propels real-world beyond-the-view vision-language navigation. *arXiv preprint arXiv:2602.05827*, 2026.

Junwei Zhou and Yu-Wing Tai. Gena3d: Generative amodal 3d modeling by bridging 2d priors and 3d coherence. *arXiv preprint arXiv:2511.21945*, 2025.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX

A METHOD DETAILS

This section provides implementation details for *GEM3D Pretraining* , complementing the pipeline overview in Fig. 2.

A.1 OBJECTIVE DETAILS

The implementation stores the voxel feature as $F_t \in \mathbb{R}^{B\times C\times D\times D\times D}$ and flattens its spatial axes to the $D^3 \times C$ notation used in the main paper. All target depth maps are back-projected to the shared world frame and cropped to the task workspace before constructing P_{full}.

Multi-scale reconstruction. Let $n_i = |\hat{P}_i|$ and $n_{\max} = \max_i n_i$. We sample $\bar{P} = \text{FPS}(P_{\text{full}}, n_{\max})$ once and use $P_i = \bar{P}_{1:n_i}$ as the target at scale i. The implementation uses the symmetric Chamfer-

$$\text{CD}(A,B) = \frac{1}{2}\left(\frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|_2 + \frac{1}{|B|}\sum_{b\in B}\min_{a\in A}\|b-a\|_2\right), \tag{A.1}$$

with scale weights $(w_0, w_1, w_2, w_3) = (2.5, 1.5, 1.25, 2.0)$. The distances are not squared.

Gaussian rendering. For anchor $\hat{p}_j$, the Gaussian head predicts center, opacity, anisotropic scale, rotation, and spherical-harmonic color as

$$\mu_j = \hat{p}_j + \eta\Delta\mu_j, \quad \alpha_j = \text{sigmoid}(o_j), \quad s_j = \exp(\ell_j), \quad q_j = r_j/\|r_j\|_2. \tag{A.2}$$

For an image pair (X, Y), the focal-reweighted MSE is

$$\ell_{\text{fMSE}}(X,Y) = \text{mean}\left[\left(1-e^{-|X-Y|}\right)^2 (X-Y)^2\right]. \tag{A.3}$$

For each valid target view, it is applied to both the full RGB image and the depth-valid region $M_v = \mathbf{1}[D_v > 0]$. Masked LPIPS and DSSIM terms are added after rendering starts; padded views are excluded from all reductions. During the geometry-only warm-up, $\mathcal{L}_{\text{rend}} = 0$.

Policy distillation and action loss. The frozen teacher is evaluated without gradients. A 5^3 patchification with stride 5 converts each 100^3 voxel feature into 8,000 patch tokens, while adaptive max pooling produces one global token. Let (p_t, g_t) and $(\bar{p}_{t+1}, \bar{g}_{t+1})$ be the current and predicted-next policy tokens, and $(\tilde{p}_t, \tilde{g}_t)$ the corresponding teacher tokens. With $\mathcal{L}_{\cos}(x,y) = 1 - \text{mean}(\text{cosine}(x,y))$, the expanded multi-step objective is

$$\begin{aligned}\mathcal{L}_{\text{multi-distill}} = {} & \mathcal{L}_{\cos}(p_t, \text{sg}[\tilde{p}_t]) + \mathcal{L}_{\cos}(g_t, \text{sg}[\tilde{g}_t]) \\ & + \mathcal{L}_{\cos}(\bar{p}_{t+1}, \text{sg}[\tilde{p}_{t+1}]) + \mathcal{L}_{\cos}(\bar{g}_{t+1}, \text{sg}[\tilde{g}_{t+1}]).\end{aligned} \tag{A.4}$$

The action loss is the sum of cross-entropy losses for translation, three discretized rotation axes, gripper state, and collision state:

$$\mathcal{L}_{\text{action}} = \mathcal{L}_{\text{CE}}^{\text{trans}} + \sum_{k\in\{x,y,z\}} \mathcal{L}_{\text{CE}}^{\text{rot},k} + \mathcal{L}_{\text{CE}}^{\text{grip}} + \mathcal{L}_{\text{CE}}^{\text{collision}}. \tag{A.5}$$

All action terms use unit weight. The distillation coefficient follows the configured decreasing schedule $[1.0, 0.5, 0.3, 0.2, 0.1, 0.05]$.

A.2 IMPLEMENTATION DETAILS OF *3D Feature Extractor*

Given a single-view observation $o^t = \{\mathcal{I}^t,\ \mathcal{D}^t,\ \mathcal{K}^t\}$ at timestep t, the camera parameters $\mathcal{K}^t$ include a camera-to-world extrinsic matrix $E_{\text{c2w}} = \begin{bmatrix} R & t \\ \mathbf{0}^\intercal & 1 \end{bmatrix}$ and the intrinsic matrix $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$.

The extrinsic maps homogeneous camera coordinates to the shared world frame. For datasets that store a world-to-camera pose, we invert it before back-projection.

The depth map $\mathcal{D}^t$ is back-projected to 3D in two steps. First, each pixel (u, v) is lifted into the camera coordinate system using the intrinsics K:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} \frac{(u-c_x)\,\mathcal{D}^t(u,v)}{f_x} \\ \frac{(v-c_y)\,\mathcal{D}^t(u,v)}{f_y} \\ \mathcal{D}^t(u,v) \end{bmatrix}. \tag{A.6}$$

The resulting point is transformed into the shared world coordinate system in homogeneous coordinates:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = E_{\text{c2w}} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}. \tag{A.7}$$

Point clouds back-projected from depth maps typically contain redundant observations, such as background and floor points. We therefore first **crop** the raw point cloud and apply **farthest point sampling (FPS)** to regularize its spatial density and obtain a fixed number of points for batched training. The processed point cloud is denoted as $\mathcal{P}$.

However, point clouds are inherently unstructured, which makes global reasoning and occlusion-aware completion difficult. To obtain a structured spatial representation, we introduce a dense volumetric intermediate representation. Specifically, each point in $\mathcal{P}$ is associated with its RGB value and the corresponding DinoV2 feature extracted from the input image. The resulting occupancy and feature-augmented point clouds are jointly voxelized to produce two volumetric observations: a ***3D Occupancy Map*** and a ***3D Feature Map*** (see Fig. A.1).

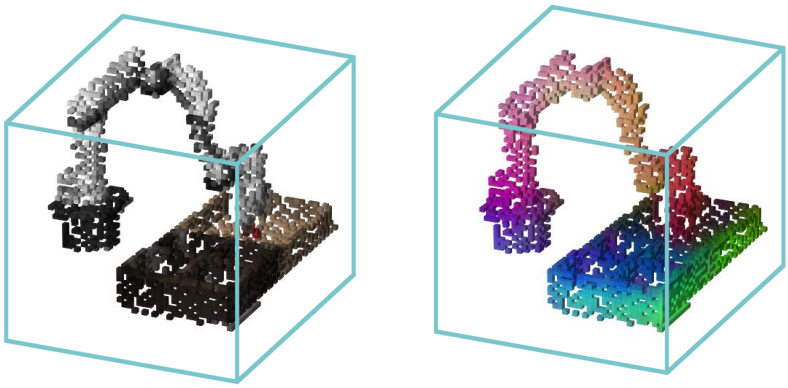


Figure A.1. Volumetric observation of the single-view input o^t. The left is the ***3D Occupancy Map*** of size $(d^3, 10)$, and the right is the ***3D Feature Map*** of size $(d^3, 384)$.

Feature extraction is performed in the voxelized space using a **fusion-style 3D U-Net**, which combines the ***3D Occupancy Map*** and the ***3D Feature Map*** to produce a dense volumetric feature $\mathcal{F}$ of size $(D^3, 128)$.

The overall procedure is summarized in Algorithm 1.

A.3 Implementation Details of *3D Deformable Attention*

In our **coarse-to-fine point cloud reconstruction** module, we adopt a **3D Deformable Cross-Attention (DCA)** mechanism operating over voxel proposal queries. Specifically, the input x corresponds to the *proposal voxel queries* $\mathcal{Q}_p \in \mathbb{R}^{N_q \times C}$ from the preceding stage, where $N_q = d^3$ is the

Algorithm 1: 3D Feature Extractor

Input: $\mathcal{I}^t, \mathcal{D}^t, \mathcal{K}^t = \{E_{\text{c2w}}, K\}$
Output: Dense volumetric feature $\mathcal{F}$
$\mathcal{F}_{2D} \leftarrow \texttt{DinoV2}(\mathcal{I}^t)$ // $(H \times W, 384)$
$P_{\text{raw}} \leftarrow \textsc{Pixel2Point}(\mathcal{D}^t, \mathcal{K}^t)$ // Eq. A.6, Eq. A.7
$P_{\text{crop}} \leftarrow \textsc{Crop}(P_{\text{raw}})$
$P \leftarrow \text{FPS}(P_{\text{crop}}, N)$
$(V_{\text{occ}}, V_{\text{feat}}) \leftarrow \textsc{Voxelize}(P, \mathcal{I}^t, \mathcal{F}_{2D})$ // $(D^3, 10)$, $(D^3, 384)$
$\mathcal{F} \leftarrow \texttt{3D UNet}(V_{\text{occ}}, V_{\text{feat}})$ // $(D^3, 128)$
return $\mathcal{F}$

number of query voxels and C is the channel dimension (here $C = 128$). We denote the number of sampled offsets per query by K. The context features are provided by the dense volumetric feature $\mathcal{F}$ extracted from the fusion-style 3D U-Net.

For each voxel query token $\mathbf{q}_p \in \mathcal{Q}_p$, the module predicts a set of learnable offsets $\{\Delta\mathbf{p}_k(\mathbf{q}_p)\}_{k=1}^K$ around a reference voxel position $\mathbf{p}_0$, forming a *voxel-wise sampling grid* v_{grid}. Keys and values are sampled from $\mathcal{F}$ at these locations using `F.grid_sample`. The sampled values are then aggregated with attention weights $A_k(\mathbf{q}_p)$ to capture local geometric context. The refined token feature, referred to as the *seed token*, is obtained as:

$$\text{DCA}(\mathbf{q}_p, \mathcal{F}) = \sum_{k=1}^{K} \tilde{A}_k(\mathbf{q}_p)\, \mathcal{F}(\mathbf{p}_0 + \Delta\mathbf{p}_k(\mathbf{q}_p)). \tag{A.8}$$

To enhance 3D geometric awareness, a **continuous positional bias (CPB)** is integrated. For each query voxel token $\mathbf{q}_p$ and its sampled locations $\mathbf{p}_0 + \Delta\mathbf{p}_k(\mathbf{q}_p)$, CPB produces a learnable bias

$$b_{p,k} = \text{CPB}\big(\mathbf{q}_p, \mathbf{p}_0 + \Delta\mathbf{p}_k(\mathbf{q}_p)\big), \tag{A.9}$$

which is directly added to the raw attention logits $e_k(\mathbf{q}_p)$ before normalization:

$$\tilde{A}_k(\mathbf{q}_p) = \text{softmax}_k\big(e_k(\mathbf{q}_p) + b_{p,k}\big), \tag{A.10}$$

Together with the learned offsets, this design allows DCA to selectively aggregate features from spatially relevant voxels and refine each voxel token with local geometric context. In essence, CPB provides a learnable bias that modulates attention weights based on relative 3D positions.

Compared to standard full-grid 3D cross-attention, which has complexity $O(N_q D^3 C)$ for N_q queries and D^3 voxels, DCA reduces the attention cost to $O(N_q K C)$ by attending to only $K \ll D^3$ sampled locations per query.

Algorithm 2 summarizes the implementation procedure of 3D DCA.

Algorithm 2: 3D Deformable Cross-Attention

Input: Voxel proposal queries $\mathcal{Q}_p \in \mathbb{R}^{N_q \times C}$, dense volumetric feature $\mathcal{F}$, number of offsets K
Output: Refined voxel tokens $\mathcal{Q}_p^{\text{refined}}$
$\{\Delta\mathbf{p}_k(\mathbf{q}_p)\}_{k=1}^K \leftarrow \textsc{OffsetPredictor}(\mathcal{Q}_p)$ // Learnable offsets per voxel token
$\mathbf{p}_k \leftarrow \mathbf{p}_0 + \Delta\mathbf{p}_k(\mathbf{q}_p)$ // Sampling locations in 3D voxel space
$\mathsf{K}, \mathsf{V} \leftarrow \texttt{F.grid_sample}(\mathcal{F}, \mathbf{p}_k)$ // Extract voxel features at sampled offsets
$\tilde{A}_k(\mathbf{q}_p) \leftarrow \textsc{Softmax}(\langle \mathbf{q}_p, \mathsf{K} \rangle + \text{CPB}(\mathbf{q}_p, \mathbf{p}_k))$ // Continuous positional bias modulates attention
$\mathcal{Q}_p^{\text{refined}} \leftarrow \sum_{k=1}^K \tilde{A}_k(\mathbf{q}_p)\, \mathsf{V}_k$ // Seed tokens capturing local geometry
return $\mathcal{Q}_p^{\text{refined}}$

Design Highlights of 3D DCA:

- Adaptive offsets enable attention to focus on locally informative voxels rather than uniform grids.
- Dense voxel features are queried via the voxel-wise sampling grid using `F.grid_sample`, enabling efficient aggregation of local geometric information from the volumetric feature field to each proposal voxel query.
- Continuous positional bias (CPB) encodes relative 3D positions and guides attention toward geometrically relevant locations.

B Training Details

B.1 Training Details for ***GEM3D Pretraining***

Training on robotic scenes with large appearance, geometry, and motion variation is challenging. The reconstruction visualizations and ablations in Fig. 5, Fig. 6, and Fig. 7 show that stable geometry and rendering quality are important for downstream policy learning. In practice, we adopt two training strategies to stabilize feed-forward Gaussian Splatting across diverse scenes:

- *Stage-wise decoupled reconstruction and rendering.* The quality of Gaussian Splatting reconstruction is highly sensitive to the initialization of point positions. Conventional 3D Gaussian Splatting often relies on point clouds initialized from COLMAP. Similarly, during training, the rendering stage (c) strongly depends on the reconstruction quality of stage (b), especially when both modules share the same feature representation $\mathcal{F}$ to simultaneously solve two related but distinct tasks: geometry reconstruction and novel view synthesis.

  If novel view rendering is enforced too early while the reconstruction quality is still poor, the gradients from stage (c) may negatively interfere with the learning of $\mathcal{F}$, leading to unstable convergence and even complete collapse of the reconstructed scene. Therefore, during the first δ training iterations, we optimize only the reconstruction objective of stage (b) and disable the rendering module (c). After the model can recover reliable and appearance-consistent point clouds from $\mathcal{F}$, we enable stage (c) and allow gradients from novel-view rendering to be back-propagated. This strategy provides a stable geometric prior for subsequent Gaussian Splatting.
- *Multi-step iterative reconstruction per scene.* A fundamental difference between feed-forward Gaussian Splatting and traditional optimization-based splatting lies in the need for strong cross-scene generalization. In the early training phase, directly switching to a new scene after a single gradient update often leads to reconstruction failure due to large scene variance.

  To mitigate this issue, we perform k consecutive optimization steps on the same scene before switching to the next one. As training progresses and the model becomes more stable, we gradually decrease k until it eventually becomes 1, corresponding to standard single-step updates per scene.

B.1.1 Training Strategy for ***GEM3D Policy***

During policy learning, imitation is performed from human demonstrations. Raw trajectories often contain redundant intermediate states, such as gripper approaching motions. We therefore select a sparse set of keyframes that correspond to interaction-critical states.

Given a demonstration trajectory

$$\mathcal{D} = \{o_t\}_{t=1}^{T},$$

where $g_t \in \{0, 1\}$ denotes the gripper open/close state, a timestep t is selected as a keyframe if

$$t \in \mathcal{K} \iff (g_t \neq g_{t-1}) \vee (t = T) \vee \textsc{Stopped}(o_t), \tag{B.11}$$

where $\textsc{Stopped}(\cdot)$ indicates near-static motion under a small velocity threshold. Redundant adjacent keyframes are further removed. This strategy shortens the demonstration sequence while preserving states that are critical for object interaction.

Table B.1. Key Hyperparameters of ***GEM3D Pretraining***

Parameters	Value
`train.batch_size`	1×8 GPUs
`train.learning_rate`	1.0×10^{-4}
`train.optimizer`	torch.optim.AdamW
`train.steps`	50K
`train.num_views`	8
`train.render_start_step`	4K
`train.`k	[4, 3, 2, 1]
`train.demos`	50
`network.`D	100
`network.`d	[7,7,7]
`network.fps_sample_num`	6144
`network.scene_bounds`	[-0.375,-0.5,0.6,1.0,0.5,1.6]

Table B.2. Key Hyperparameters of ***GEM3D Policy***

Parameters	Value
`train.batch_size`	1
`train.learning_rate`	1.0×10^{-4}
`train.optimizer`	torch.optim.AdamW
`train.single_task_steps`	12K
`train.camera`	[front]
`train.demos`	20
`network.scene_bounds`	[-0.375,-0.5,0.6,1.0,0.5,1.6]
`network.num_tokens`	8000
`network.num_latents`	2048
`network.`$\lambda_d istill$	[1,0.5,0.3,0.2,0.1,0.05]

B.2 Parameter Details

C Task Descriptions

We selected 12 tasks from RLBench, which together contain 178 variations. Fig. C.2 and Table C.3 provide detailed descriptions.

Notably, among these 12 tasks, `close jar` and `open jar`, `meat off grill` and `meat on grill`, and `put money in safe` and `take money out safe` share the same scene but require different action trajectories. Thus, the 12 tasks cover nine distinct scenes. During pretraining, we collect data from nine tasks to cover these scenes, while reserving several action trajectories for downstream policy learning. This design evaluates whether *GEM3D* can reuse pretrained scene-level geometry for new manipulation trajectories within related environments.

Specifically, the trajectories for `open jar`, `meat on grill`, and `take money out safe` were not seen during pretraining. Nevertheless, the in-domain results in Table D.4 show that *GEM3D* still improves performance on these extrapolated tasks. The view-generalization results in Table D.6 further show that our method remains more robust when the test camera changes, even for tasks with different trajectories within the same scene. This indicates that single-view expert demonstrations during policy learning are sufficient to adapt the pretrained representation to new skills, while preserving scene-level geometric understanding.

D Supplementary Experiments

D.1 Full In-domain RLBench Results

Table D.4 reports the complete in-domain results on all 12 RLBench tasks. For compactness, the main text shows six representative task columns while computing the average success rate over

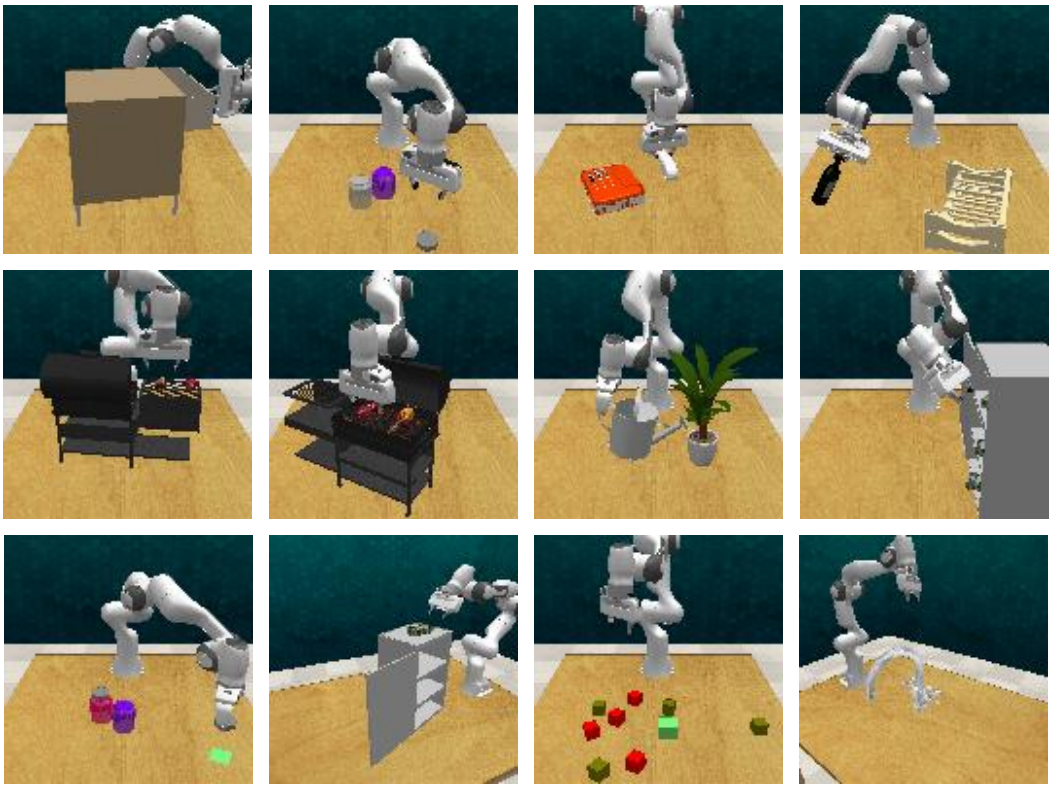


Figure C.2. RLBench tasks. 12 tasks from nine scenes are selected.

Table C.3. Task Descriptions. The 12 highly challenging tasks include a variety of random variations in color and layout, as well as multiple language instruction variations.

Task	Type	Variations	Instruction Template
`Close Jar`	color	20	*"close the _ jar"*
`Open Drawer`	placement	3	*"open the _ drawer"*
`Meat off Grill`	size	2	*"take the _ off the grill"*
`Turn Tap`	category	2	*"turn _ tap"*
`Water Plants`	placement	1	*"water plant"*
`Phone on Base`	placement	1	*"put the phone on the base"*
`Put Money in Safe`	placement	3	*"put the money away in the safe on the _ shelf"*
`Place Wine`	placement	1	*"stack wine bottle"*
`Stack Blocks`	color, count	60	*"stack _ _ blocks"*
`Open Jar`	color	80	*"open the _ jar"*
`Meat on Grill`	category	2	*"put the _ on the grill"*
`Take Money out Safe`	placement	3	*"take the money out of the _ shelf and place it on the table."*

all 12 tasks. Across the full benchmark, *GEM3D* achieves the best average SR (**44.2%**) and the best average rank, improving over the strongest baseline, *Spatial Forcing* (35.1%), by **9.1** points. The gains are especially clear on extrapolated trajectories such as `meat on grill` and `take money out safe`, where the task shares the scene with pretraining data but requires a different action sequence.

D.2 Inference Efficiency on RLBench

Table D.5 reports policy inference speed on the RLBench `open drawer` task using a single NVIDIA RTX 3090: *GEM3D* achieves 1.28 Hz, compared with 1.18 Hz for *PerAct* and 1.36 Hz for *ManiGaussian*. At inference, the policy predicts actions from calibrated single-view RGB-D observations without explicit point-cloud reconstruction or Gaussian rendering. Distillation adds

Table D.4. Full in-domain results on all 12 RLBench tasks using the training view (SR%). **Bold** numbers denote the best performance, and underlined numbers denote the second best.

Methods	Avg. SR (%) ↑	Avg. Rank ↓	Close Jar	Open Drawer	Meat off Grill	Turn Tap	Water Plants
PerAct	21.9 ± 5.3	4.58	19.0 ± 4.6	38.7 ± 13.7	28.3 ± 4.7	43.7 ± 4.7	8.3 ± 2.3
GNFactor	29.2 ± 18.4	3.33	32.0 ± 16.8	51.7 ± 13.5	32.0 ± 10.5	40.7 ± 10.7	22.0 ± 4.4
ManiGaussian	31.5 ± 8.5	3.08	27.3 ± 16.4	57.3 ± 1.5	30.0 ± 11.3	44.0 ± 10.5	19.3 ± 9.0
ManiGaussian (w Distill)	33.3 ± 10.4	3.00	31.0 ± 12.1	54.3 ± 12.5	35.3 ± 9.8	42.7 ± 14.6	16.3 ± 8.7
DP3	33.0 ± 7.6	4.17	29.7 ± 8.6	33.4 ± 7.3	50.2 ± 9.1	46.8 ± 8.7	21.6 ± 6.4
Spatial Forcing	35.1 ± 7.9	3.17	31.4 ± 9.4	55.0 ± 9.6	38.2 ± 8.4	46.5 ± 9.0	21.3 ± 6.3
***GEM3D** (Ours)*	**44.2** ± 6.2	**1.00**	**44.7** ± 3.2	**65.7** ± 4.5	**44.3** ± 3.1	**54.0** ± 7.2	**32.7** ± 3.5

Methods	Phone on Base	Put Money in Safe	Place Wine	Stack Blocks	Open Jar	Meat on Grill	Take Money out Safe
PerAct	51.3 ± 11.9	10.7 ± 0.6	3.0 ± 1.0	4.0 ± 0.0	26.7 ± 9.1	16.0 ± 7.9	13.7 ± 2.9
GNFactor	62.0 ± 5.6	11.0 ± 10.5	5.7 ± 2.1	3.7 ± 1.2	35.7 ± 6.4	35.0 ± 6.1	19.0 ± 4.0
ManiGaussian	66.3 ± 12.5	5.7 ± 3.5	8.0 ± 2.6	2.0 ± 1.7	38.3 ± 9.0	30.7 ± 12.9	49.3 ± 10.5
ManiGaussian (w Distill)	64.3 ± 16.2	20.3 ± 9.2	5.3 ± 3.5	5.0 ± 2.6	33.7 ± 9.5	48.7 ± 2.9	43.0 ± 23.4
DP3	63.7 ± 9.8	15.4 ± 4.6	12.7 ± 3.8	8.3 ± 2.7	26.8 ± 6.9	55.5 ± 8.4	31.6 ± 7.1
Spatial Forcing	63.2 ± 10.2	24.1 ± 7.3	7.4 ± 3.8	4.6 ± 2.9	35.6 ± 8.7	47.1 ± 8.1	46.9 ± 10.1
***GEM3D** (Ours)*	**67.3** ± 1.5	**29.7** ± 2.9	**11.7** ± 4.0	**7.7** ± 2.9	**40.0** ± 5.6	**61.0** ± 22.1	**72.0** ± 13.7

Table D.5. Policy inference speed and training cost on the RLBench `open drawer` task. All measurements are taken on a single NVIDIA RTX 3090.

Method	Inference Rate	Policy Training Cost
PerAct	1.18 Hz	2–3 h
ManiGaussian	1.36 Hz	4–5 h
GEM3D Policy	1.28 Hz	5–6 h

computation during policy training, while the frozen teacher and reconstruction branches are not used during deployment.

D.3 View Generalization and Occlusion-sensitive Tasks

Table D.6 extends the view-generalization evaluation to eight RLBench tasks, including two heavily occluded tasks, `open drawer` and `take money out safe`. *GEM3D* remains the strongest method on average under both moderate and large viewpoint shifts. The performance drop is more pronounced on the two occlusion-sensitive tasks, especially under large shifts, because one side of the scene can become almost fully hidden from the single RGB-D camera, as illustrated in Fig. D.3. This indicates that visibility of task-critical contact regions still matters, even when the policy has learned a stronger scene-complete representation.

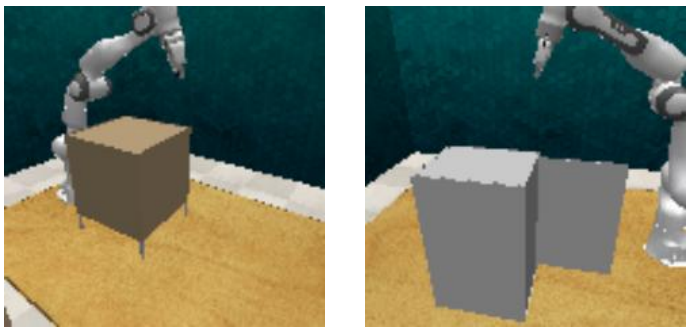
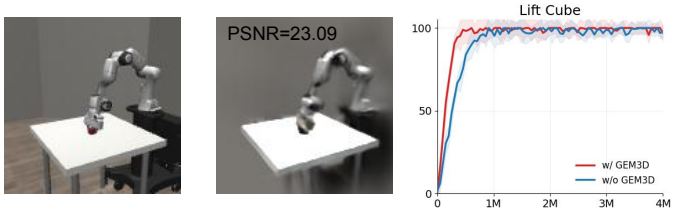


Figure D.3. `open drawer` and `take money out safe` tasks under large view shifts. Both tasks suffer from severe occlusion, resulting in drastic success rate drops.

Table D.6. Results on eight RLBench tasks at the training view and under moderate/large viewpoint shifts (SR%). Averages use the same eight tasks in all settings. **Bold** numbers denote the best performance, and ↓ indicates the relative performance drop compared to the training view.

	Methods	Close Jar	Meat off Grill	Turn Tap	Phone on Base	Open Jar	Meat on Grill	Open Drawer	Take Money out Safe	Avg.
Train View	*PerAct*	19.0 ± 4.6	28.3 ± 4.7	43.7 ± 4.7	51.3 ± 11.9	26.7 ± 9.1	16.0 ± 7.9	38.7 ± 13.7	13.7 ± 2.9	29.7 ± 13.7
	ManiGaussian	27.3 ± 16.4	30.0 ± 11.3	44.0 ± 10.5	66.3 ± 12.5	38.3 ± 9.0	30.7 ± 12.9	57.3 ± 1.5	49.3 ± 10.5	42.9 ± 14.0
	ManiGaussian (w Distill)	31.0 ± 12.1	35.3 ± 9.8	42.7 ± 14.6	64.3 ± 16.2	33.7 ± 9.5	48.7 ± 2.9	54.3 ± 12.5	43.0 ± 23.4	44.1 ± 11.3
	DP3	29.7 ± 8.6	50.2 ± 9.1	46.8 ± 8.7	63.7 ± 9.8	26.8 ± 6.9	55.5 ± 8.4	33.4 ± 7.3	31.6 ± 7.1	42.2 ± 8.1
	Spatial Forcing	31.4 ± 9.4	38.2 ± 8.4	46.5 ± 9.0	63.2 ± 10.2	35.6 ± 8.7	47.1 ± 8.1	55.0 ± 9.6	46.9 ± 10.1	45.5 ± 8.9
	GEM3D (ours)	**44.7** ± 3.2	**44.3** ± 3.1	**54.0** ± 7.2	**67.3** ± 1.5	**40.0** ± 5.6	**61.0** ± 22.1	**65.7** ± 4.5	**72.0** ± 13.7	**56.1** ± 12.1
Moderate Shift	*PerAct*	7.7 ± 9.9 (59.65%↓)	19.7 ± 7.2 (30.56%↓)	14.0 ± 7.2 (67.94%↓)	0.3 ± 0.6 (99.35%↓)	2.0 ± 1.7 (92.50%↓)	12.3 ± 9.6 (22.92%↓)	19.7 ± 7.2 (49.14%↓)	3.3 ± 3.2 (75.61%↓)	9.9 ± 7.7 (62.21%↓)
	ManiGaussian	15.3 ± 15.9 (43.91%↓)	17.0 ± 12.1 (43.33%↓)	32.3 ± 2.3 (26.52%↓)	30.7 ± 18.6 (53.76%↓)	18.3 ± 2.5 (52.18%↓)	24.3 ± 10.1 (20.85%↓)	30.3 ± 18.0 (47.09%↓)	12.0 ± 4.4 (75.67%↓)	22.5 ± 7.9 (45.43%↓)
	ManiGaussian (w Distill)	20.3 ± 10.3 (34.41%↓)	23.7 ± 2.3 (33.01%↓)	34.0 ± 9.8 (20.32%↓)	40.0 ± 22.6 (37.82%↓)	21.7 ± 3.2 (35.64%↓)	42.3 ± 7.0 (13.02%↓)	31.7 ± 4.5 (41.71%↓)	21.0 ± 14.8 (51.16%↓)	29.3 ± 8.9 (33.39%↓)
	DP3	13.8 ± 6.5 (53.5%↓)	23.7 ± 7.2 (52.8%↓)	19.5 ± 6.8 (58.3%↓)	28.6 ± 8.1 (55.1%↓)	10.2 ± 5.4 (61.9%↓)	24.4 ± 7.1 (56.0%↓)	15.1 ± 6.0 (54.8%↓)	13.9 ± 5.8 (56.0%↓)	18.6 ± 6.6 (55.8%↓)
	Spatial Forcing	15.8 ± 6.8 (49.7%↓)	19.7 ± 7.3 (48.4%↓)	22.4 ± 7.1 (51.8%↓)	31.6 ± 8.5 (50.0%↓)	17.2 ± 6.6 (51.7%↓)	24.1 ± 7.4 (48.8%↓)	23.8 ± 7.6 (56.7%↓)	19.3 ± 7.0 (58.8%↓)	21.7 ± 7.3 (52.2%↓)
	GEM3D (ours)	**33.7** ± 8.5 **(24.62%↓)**	**33.0** ± 5.0 **(25.56%↓)**	**47.7** ± 8.4 **(11.67%↓)**	**48.3** ± 13.6 **(28.22%↓)**	**34.3** ± 4.0 **(14.18%↓)**	**47.0** ± 14.8 **(22.95%↓)**	**42.7** ± 9.5 **(35.02%↓)**	**41.7** ± 13.6 **(42.13%↓)**	**41.0** ± 6.5 **(25.54%↓)**
Large Shift	*PerAct*	2.8 ± 3.4 (85.11%↓)	15.7 ± 3.1 (44.69%↓)	9.7 ± 4.3 (77.86%↓)	1.0 ± 2.0 (98.05%↓)	3.3 ± 4.1 (87.51%↓)	4.8 ± 3.8 (69.81%↓)	1.5 ± 1.9 (96.12%↓)	1.2 ± 2.4 (91.44%↓)	5.0 ± 5.2 (81.32%↓)
	ManiGaussian	10.3 ± 14.2 (62.20%↓)	16.3 ± 8.4 (45.57%↓)	27.0 ± 7.0 (38.64%↓)	23.7 ± 16.0 (64.32%↓)	21.5 ± 6.9 (43.91%↓)	16.0 ± 6.2 (47.88%↓)	0.0 ± 0.0 (100.00%↓)	1.3 ± 1.8 (97.30%↓)	14.5 ± 10.0 (62.49%↓)
	ManiGaussian (w Distill)	21.5 ± 9.2 (45.16%↓)	16.3 ± 6.9 (67.92%↓)	32.0 ± 8.6 (29.69%↓)	33.7 ± 13.1 (41.97%↓)	23.2 ± 12.0 (62.38%↓)	28.2 ± 8.0 (53.42%↓)	0.7 ± 1.2 (98.77%↓)	5.3 ± 7.4 (87.60%↓)	20.1 ± 12.0 (60.86%↓)
	DP3	8.7 ± 4.7 (70.7%↓)	15.4 ± 5.8 (69.3%↓)	13.2 ± 5.3 (71.8%↓)	17.8 ± 6.2 (72.1%↓)	6.1 ± 4.1 (77.2%↓)	14.7 ± 5.6 (73.5%↓)	8.4 ± 4.9 (74.9%↓)	7.6 ± 4.5 (75.9%↓)	11.5 ± 5.1 (72.8%↓)
	Spatial Forcing	9.6 ± 5.1 (69.4%↓)	12.8 ± 5.9 (66.5%↓)	14.7 ± 5.6 (68.4%↓)	18.5 ± 6.4 (70.7%↓)	10.4 ± 5.0 (70.8%↓)	15.3 ± 5.8 (67.5%↓)	13.7 ± 5.5 (75.1%↓)	11.9 ± 5.2 (74.6%↓)	13.4 ± 5.6 (70.6%↓)
	GEM3D (ours)	**30.2** ± 7.1 **(32.46%↓)**	**26.0** ± 5.8 **(41.35%↓)**	**45.3** ± 6.3 **(16.11%↓)**	**39.0** ± 13.2 **(42.08%↓)**	**35.8** ± 7.7 **(10.42%↓)**	**40.3** ± 6.8 **(33.88%↓)**	**11.0** ± 15.0 **(83.25%↓)**	**8.2** ± 10.0 **(88.56%↓)**	**29.5** ± 13.7 **(43.53%↓)**



(a) groundtruth (b) rendering (c) Success Rate of SAC

Figure D.4. Robosuite RL transfer. GEM3D maintains high rendering fidelity in Robosuite and improves downstream RL convergence on the `Lift` task.

D.4 Rendering Quality Analysis

Table D.7 provides additional quantitative reconstruction results on three representative RLBench tasks. *GEM3D* improves the average PSNR from 16.05 to **23.13** and SSIM from 0.3838 to **0.8136**, while reducing Chamfer L_2 from 0.04511 to **0.01621**. These results are consistent with the visual comparisons in Fig. 5 and Fig. 6: *GEM3D* produces sharper novel views and denser, more complete point clouds than *ManiGaussian*.

We further provide qualitative novel-view synthesis results in Fig. D.7 and Fig. D.8. For each keyframe sequence, novel views are rendered from uniformly distributed virtual cameras and compared across different RLBench tasks.

Table D.7. Comparison of scene reconstruction quality. Larger PSNR and SSIM indicate better image fidelity, while lower Chamfer L_2 indicates higher point cloud geometric accuracy.

Methods	Tasks	PSNR↑	SSIM↑	Chamfer L_2↓
ManiGaussian	`Open Drawer`	16.24	0.3926	0.04035
	`Turn Tap`	15.84	0.3741	0.05148
	`Water Plants`	16.07	0.3846	0.04352
	Avg	16.05	0.3838	0.04511
GEM3D *(ours)*	`Open Drawer`	23.54	0.8259	0.01541
	`Turn Tap`	22.92	0.8123	0.01708
	`Water Plants`	22.93	0.8025	0.01613
	Avg	**23.13**	**0.8136**	**0.01621**

D.5 Real-world ManiGaussian Rendering

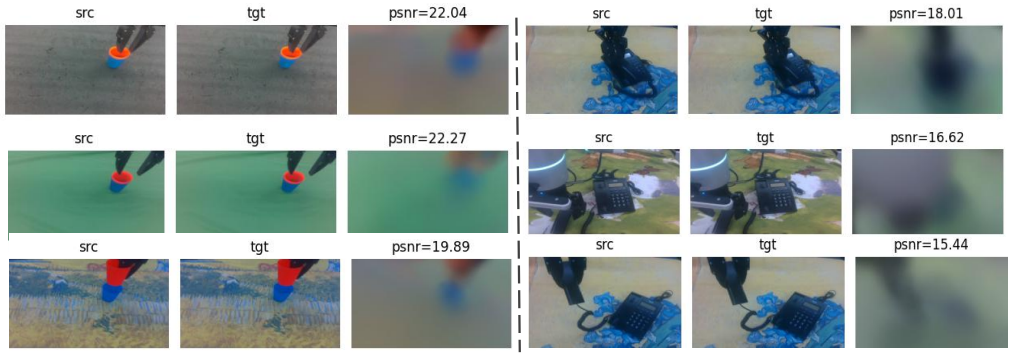


Figure D.5. Real-world rendering results of *ManiGaussian*. The source and target views are visually clear, but the rendered novel views are severely blurred and lose task-relevant object geometry.

Fig. D.5 visualizes *ManiGaussian* on real-world manipulation scenes. Although the source and target images are well aligned, the rendered views collapse into low-frequency color blobs: object boundaries are smeared, thin structures such as grippers and phone cables are missing, and contact-relevant foreground objects are often indistinguishable from the tabletop. This suggests that joint Gaussian rendering and policy learning is difficult under noisy real-world RGB-D observations. When the reconstruction branch fails to recover reliable geometry and appearance, it can provide weak or misleading supervision for action learning, which helps explain the low real-world success rates of *ManiGaussian* in the main experiments.

D.6 Multi-task Policy Learning

We also evaluate whether the learned representation supports a unified language-conditioned policy trained on multiple tasks. Following the multi-task setting used in our earlier experiments, we train a single policy on a mixture of four RLBench tasks: `open jar`, `meat on grill`, `turn tap`, and `phone on base`. Table D.8 reports the averaged multi-task results, and Table D.9 reports per-task success rates together with relative changes from the corresponding single-task setting.

Compared with *PerAct*, *GEM3D* maintains substantially higher success rates under all viewpoint settings. Its average drops from the single-task setting are small under moderate and large shifts, showing that the pretrained geometric representation and distillation-based policy learning remain effective when one policy must distinguish multiple language-conditioned skills.

Table D.8. Multi-task results on four RLBench tasks (SR%). **Bold** numbers denote the best performance, and ↓ indicates the relative performance drop compared to the single-task training setting.

Methods	Training View	Moderate Shift	Large Shift
PerAct	26.1 ± 17.5 (24.1%↓)	6.4 ± 4.5 (11.1%↓)	5.6 ± 3.6 (16.1%↑)
ManiGaussian *(w Distill)*	47.1 ± 5.8 (0.4%↓)	30.7 ± 8.1 (3.8%↓)	26.5 ± 7.7 (10.7%↓)
GEM3D *(ours)*	**52.9 ± 6.4 (1.7%↑)**	**41.2 ± 9.6 (6.9%↓)**	**36.4 ± 9.0 (3.7%↓)**

Table D.9. Results of multi-task view generalization on RLBench tasks. We report the success rate (mean ± std %) in the multi-task setting. **Bold** numbers denote the best performance. The percentage in parentheses indicates the relative performance change compared to the single-task baseline ($\frac{\text{Multi}-\text{Single}}{\text{Single}}$).

	Methods	Open Jar	Meat on Grill	Turn Tap	Phone on Base	**Avg.**
Moderate Shift	*PerAct*	4.3 ± 4.0 (115.00%↑)	2.0 ± 2.6 (83.74%↓)	15.7 ± 5.0 (12.14%↑)	3.7 ± 6.4 (1133.33%↑)	6.4 ± 4.5 (11.11%↓)
	ManiGaussian *(w Distill)*	21.0 ± 7.9 (3.23%↓)	28.7 ± 10.1 (32.15%↓)	44.3 ± 9.1 (30.29%↑)	28.7 ± 5.1 (28.25%↓)	30.7 ± 8.1 (11.01%↓)
	GEM3D *(ours)*	**27.0 ± 13.0 (21.28%↓)**	**38.7 ± 14.5 (17.66%↓)**	**59.0 ± 4.6 (23.69%↑)**	**40.0 ± 6.2 (17.18%↓)**	**41.2 ± 9.6 (7.00%↓)**
Large Shift	*PerAct*	5.3 ± 5.2 (60.61%↑)	1.2 ± 1.9 (75.00%↓)	14.5 ± 5.4 (49.48%↑)	1.5 ± 1.8 (50.00%↑)	5.6 ± 3.6 (19.15%↑)
	ManiGaussian *(w Distill)*	22.2 ± 7.2 (16.23%↓)	26.3 ± 9.3 (6.05%↑)	36.0 ± 10.2 (20.00%↑)	21.3 ± 5.7 (42.90%↓)	26.5 ± 8.1 (10.77%↓)
	GEM3D *(ours)*	**27.0 ± 10.7 (9.40%↓)**	**31.3 ± 7.4 (15.86%↓)**	**47.0 ± 12.6 (3.75%↑)**	**40.3 ± 5.4 (3.33%↑)**	**36.4 ± 9.0 (3.70%↓)**

D.7 ROBOSUITE REINFORCEMENT LEARNING TRANSFER

To evaluate whether the pretrained representation can also benefit reinforcement learning, we conduct an additional experiment on the `Lift` task in Robosuite (Zhu et al., 2020). Robosuite is a modular robot learning benchmark built on MuJoCo and provides controlled manipulation environments with configurable objects, cameras, and reward functions. In this experiment, the robot must lift a cube from the tabletop. We first pretrain the GEM3D visual encoder on an offline Robosuite dataset, then initialize a SAC agent with the distilled policy encoder for downstream RL training.

Fig. D.4 shows that *GEM3D* reconstructs task-relevant cube and robot geometry in Robosuite, while its initialization yields faster and more stable SAC learning than training without the representation. This transfer indicates that the learned 3D features also benefit reinforcement learning beyond imitation learning.

D.8 Cross-view Representation Alignment

Paired feature construction. For each of N paired `open drawer` states with the same robot–object configuration, we extract a single-view feature h_i^{s} from training view $\mathbf{c}$. Its multi-view counterpart h_i^{m} is produced by concatenating all calibrated point clouds in the shared world frame before applying the same 3D encoder. The multi-view input exposes more of the scene geometry and therefore serves as a reference for the scene structure recoverable from complete observations. We extract both feature sets from the pretrained teacher (original) and the distilled policy. If a partial single view preserves the state-to-state organization of the multi-view reference, the representation is less tied to view-specific appearance and more consistent with amodal scene understanding.

Joint MDS projection. For each representation family, we stack the four ordered feature sets—teacher single-view, teacher multi-view, policy single-view, and policy multi-view—into $H \in \mathbb{R}^{4N\times d}$ and ℓ_2-normalize every row. We form the cosine-distance matrix $D_{uv} = 1 - \hat{h}_u^\top \hat{h}_v$ and apply classical metric multidimensional scaling (MDS). With $J = I_{4N} - \frac{1}{4N}\mathbf{1}\mathbf{1}^\top$, MDS double-centers the squared distances as

$$B = -\frac{1}{2}J(D \odot D)J = V\Lambda V^\top, \qquad Z = V_{1:2}\Lambda_{1:2}^{1/2} \in \mathbb{R}^{4N\times 2}, \tag{D.12}$$

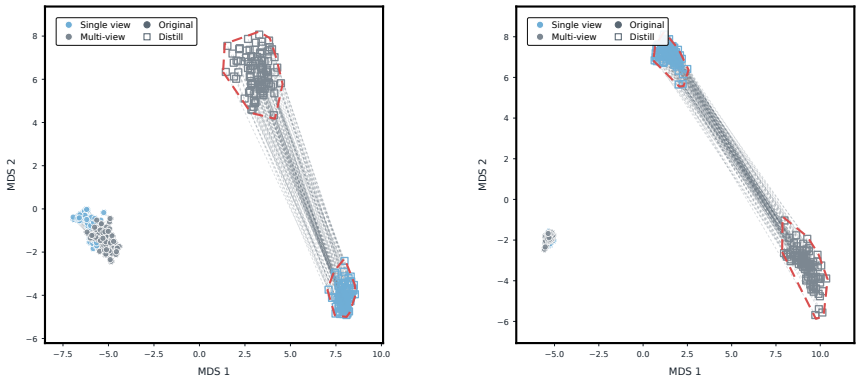
where the two largest positive eigenvalues and their eigenvectors define the plotted coordinates. Unlike neighborhood-focused stochastic projections such as t-SNE, classical MDS gives a common deterministic embedding that approximately preserves global pairwise feature distances. The orientation and sign of the two axes are arbitrary and should not be interpreted semantically.

Alignment score and interpretation. We split Z into single- and multi-view configurations $(X_r, Y_r) \in \mathbb{R}^{N\times 2}$ for stage $r \in \{\mathrm{teacher}, \mathrm{policy}\}$ and compute

$$S_{\mathrm{MDS}}(X_r, Y_r) = \left\| \left(\frac{X_r}{\|X_r\|_F}\right)^\top \left(\frac{Y_r}{\|Y_r\|_F}\right) \right\|_*. \tag{D.13}$$

Here $\|\cdot\|_*$ is the nuclear norm, making the score equivalent to optimal orthogonal alignment and invariant to rotations or reflections of the MDS axes. It compares the relative configuration of paired scene states rather than assigning meaning to absolute 2D coordinates. Higher values indicate that the single-view representation better preserves the scene-state relations recovered from the multi-view input. All methods and stages use identical paired states.

Reading the scatter plots. Blue and gray points represent single- and multi-view features; circles and squares denote pretrained-teacher and distilled-policy features. Each gray dashed segment connects the policy embeddings of the same state under single- and multi-view inputs, while red dashed contours summarize the two policy distributions. Individual point locations, axis directions, and a single connector length are not meaningful in isolation: the score compares the global paired configuration. The more congruent *GEM3D* configurations in Fig. D.6 visually agree with its higher teacher and policy alignment scores.



(a) *GEM3D* representations. **(b)** VGGT-Ω representations.

Figure D.6. MDS visualization of cross-view representation transfer. Paired `open drawer` states are shown for the pretrained teacher and distilled policy. Colors denote input completeness; markers denote transfer stage; axes are arbitrary.

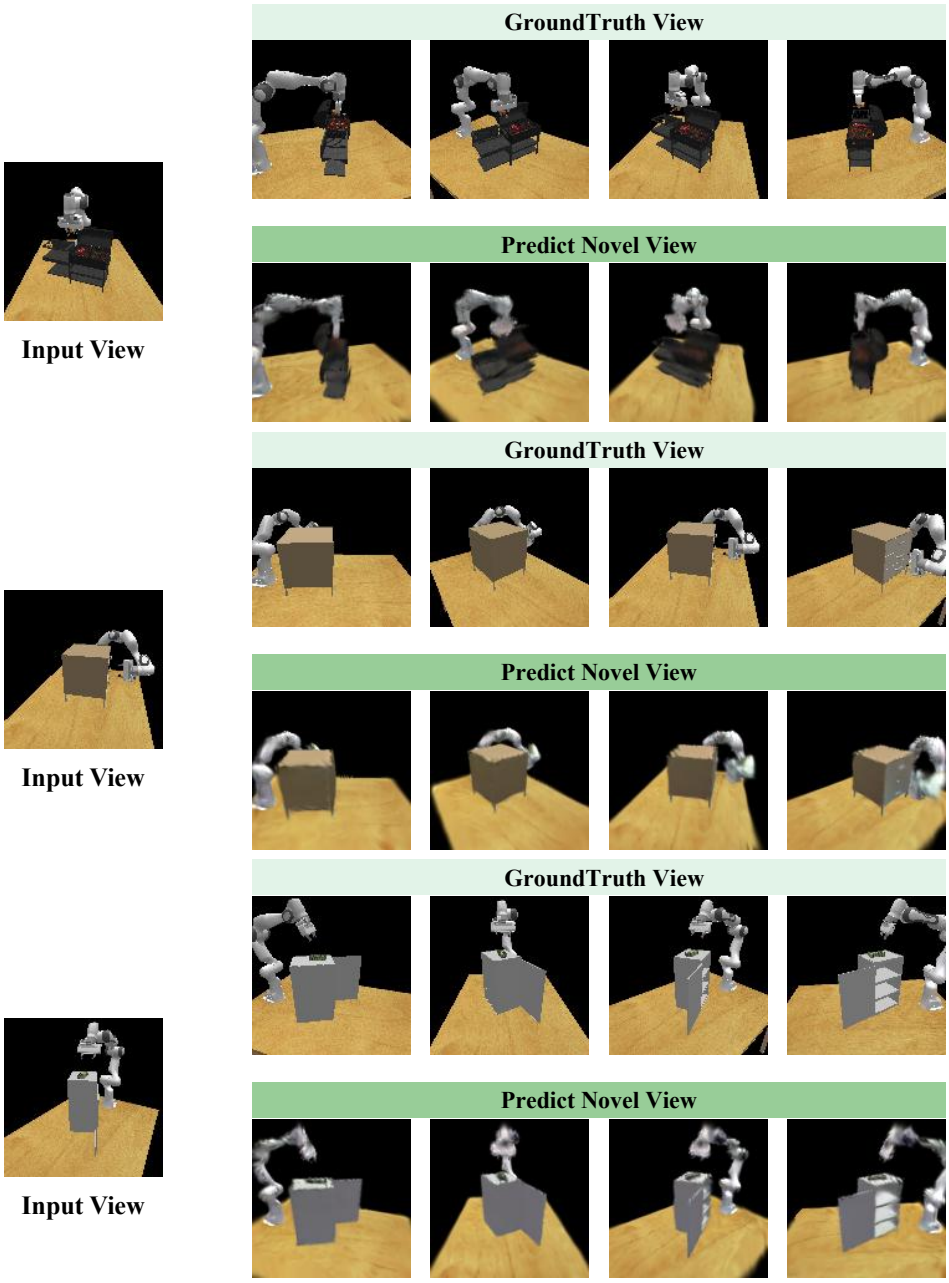


Figure D.7. Additional novel-view rendering results (Part 1/2).

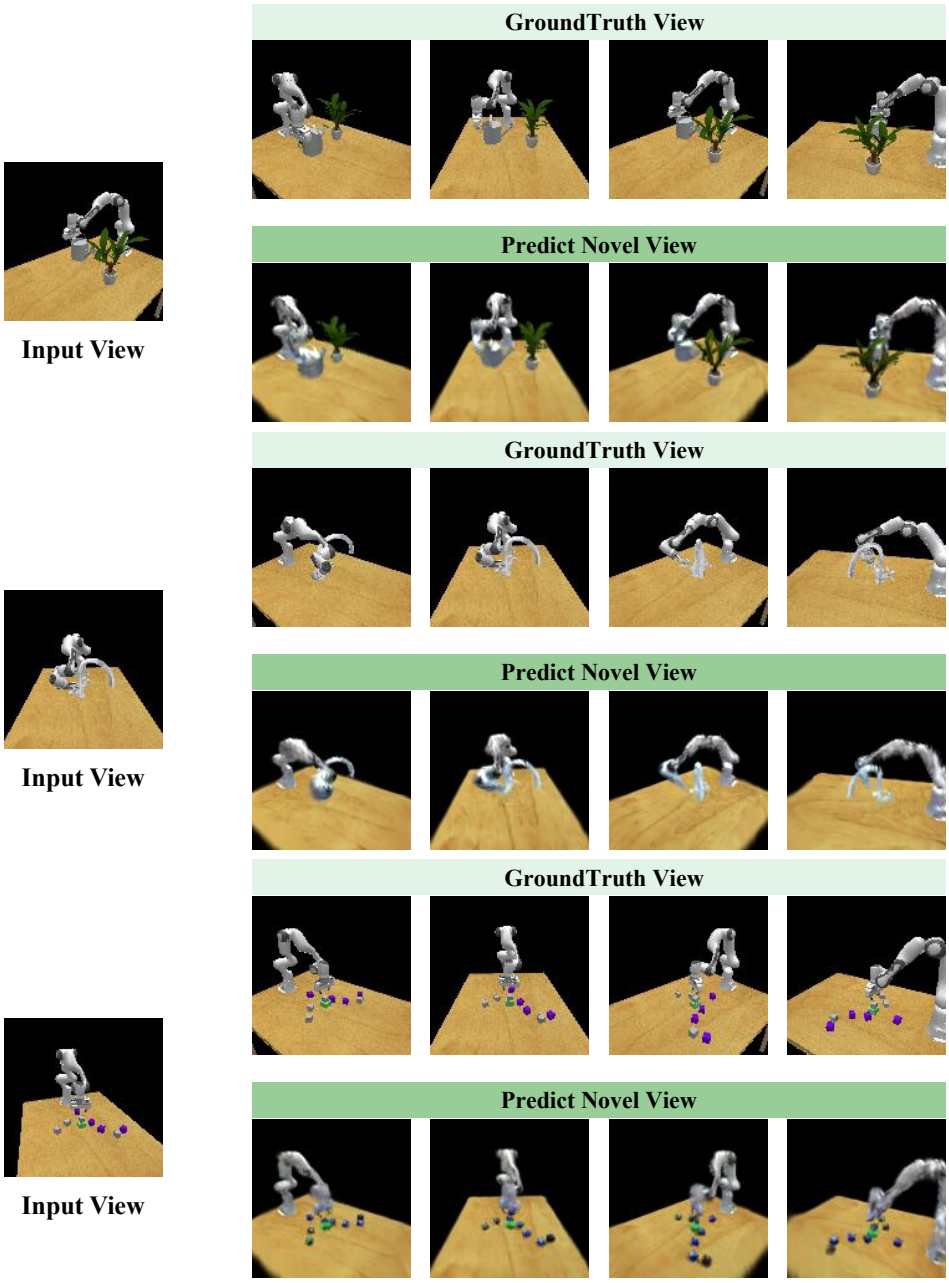


Figure D.8. Additional novel-view rendering results (Part 2/2).