2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning Structural Latent Points for Efficient Visual Representations in Robotic Manipulation

Yicheng Jiang[1,*], Jiaxu Wang[1,3,*], Junhao He[2], Zesen Gan[1], Junhao Li[1],
Qiang Zhang[2,4], Jingkai Sun[5], Jiahang Cao[5], Mingyuan Sun[6], Xiangyu Yue[3,✉] and Qiming Shao[1,✉]

***Abstract*— Current 3D-aware pretraining methods for embodied perception and manipulation are largely built on differentiable rendering frameworks, producing either fully implicit neural fields or fully explicit geometric primitives. Implicit representations, while expressive, lack explicit structural cues, whereas explicit ones preserve geometry but suffer from resolution limits and weak generalization. To address these limitations, we propose a novel pretraining framework that learns a hybrid representation-structural latent points. Specifically, we insert a point-wise latent variational autoencoder into the latent space of a point-cloud autoencoder, jointly regularizing point-wise features and coordinates toward a Gaussian prior. The resulting compact latent preserves coarse structural tendencies, which do not encode precise geometry but capture richer rough shape and semantic information, effectively combining the expressiveness of implicit representations with the structural priors of explicit ones. In addition, informed by shared design choices in prior work, we develop a streamlined, efficient 3DGS-based rendering pipeline that is deliberately kept lightweight, improving efficiency while leaving greater representational capacity to the front-end latent module. Extensive evaluations on RLBench, ManiSkill2, and a real-robot platform demonstrate consistent gains in task success, sample efficiency, and robustness to viewpoint and scene variations over strong baselines. Ablation studies further confirm that each component of our framework is critical to overall performance.**

I. Introduction

Early approaches to embodied intelligence fed raw sensory observations directly into policy networks. Although simple, this strategy was soon found to be inefficient and brittle. A more effective paradigm emerged by introducing general-purpose feature extractors that compress raw inputs into compact latent representations, improving both efficiency and downstream task performance. However, such models primarily capture 2D features and thus fall short in supporting agents that must act in the 3D physical world.

To overcome this limitation, recent works have developed various 3D-aware pretraining frameworks that exploit multi-view observations. By incorporating 3D rendering mechanisms, such as Neural Radiance Field (NeRF) [1] or 3D Gaussian Splatting (3DGS) [2] , into the training process, these methods encourage models to acquire geometry-aware representations. NeRF-based approaches typically aggregate multi-view features into a volumetric representation, which is then rendered via ray casting for self-supervised training. The resulting representation is entirely implicit, encoding spatial and semantic information in a continuous volumetric field, but without exposing any structured or explicit geometric information. While such implicit embeddings are expressive, they lack interpretable structural cues that can be directly exploited by downstream embodied agents. Moreover, NeRF-based pretraining is computationally expensive due to the inefficiency of ray casting.

In contrast, other studies leverage the high efficiency of 3DGS to build the pretraining pipeline. These methods generally predict the properties of Gaussian splats from multi-view images and adopt the resulting splats directly as explicit scene representations, sometimes further enriching each splat point with semantic features. Compared with implicit fields, such explicit point-like structures naturally encode interpretable geometric information, which can benefit downstream reasoning and manipulation tasks. However, their capacity is constrained by resolution: to maintain lightweight policy networks, point-like splats are often downsampled, leading to the loss of fine-grained structural information. To conclude, implicit methods offer strong expressiveness, while explicit methods provide well-defined structural priors.

Another challenge with the aforementioned training paradigms is that their performance often degrades when transferring to downstream tasks whose scenes differ significantly from the pretraining domain. During pretraining, the projected point clouds tend to occupy a well-structured and dense latent space, leading to strong representation quality. However, when downstream scenes deviate too much, their projections can lie far from the pretrained latent manifold, resulting in poor representations and weaker task performance.

To combine the strengths of implicit representations in expressiveness and explicit representations in structural information, we propose a framework that learns structural latent points. Specifically, our method integrates a standard point autoencoder with a point-wise latent variational autoencoder (PL-VAE). The PL-VAE operates on the sparse feature point cloud residing in the latent space of a U-Net-like point cloud autoencoder, and regularizes both the spatial distribution of points and their associated features toward a Gaussian prior. Through this process, the sparse feature points are transformed into a latent representation that remains structured yet no longer strictly explicit. Rather than encoding precise geometry, it captures structural information in a probabilistically regularized form, thereby serving as

* Contributed equally co-first authors, order determined by coin toss.
✉Corresponding authors.
[1]The Hong Kong University of Science and Technology
[2]The Hong Kong University of Science and Technology (Guangzhou)
[3]MMLab, The Chinese University of Hong Kong
[4]X-Humanoid Robots
[5]The University of Hong Kong
[6]Tsinghua University

a structural implicit representation. This design preserves geometric tendencies while retaining the compactness and flexibility of latent representations, thus bringing together the best of both paradigms.

We further adopt 3DGS to construct our self-supervised pretraining pipeline due to its computational efficiency. Unlike prior frameworks that often introduce complex and redundant architectural designs, we deliberately remove unnecessary components and retain only the core modules that are consistently shared across various approaches and proven to be effective. Our goal is to design a streamlined and lightweight pretraining pipeline that is both simple and effective. We extensively evaluate this method across RLBench and the ManiSkill2 platform and compare it with several baselines. The results show that our method achieves the best overall performance. We further validate the practicality of our pretrained framework on a real-world robotic platform across six diverse experimental tasks. Moreover, we conduct ablation studies for all necessary designs. This confirms the effectiveness of our structural latent point representation in enhancing both generalization and performance in downstream embodied tasks.

II. Related Work

A. Visual Representation Learning

Recent advances in visual representation learning have greatly influenced robot learning by enabling robots to better perceive and interact with their environments. However, developing robust visual representations for robotics remains both critical and challenging. Some methods [3], [4], [5], [6] leverage task-specific data, such as real-world robot demonstrations, but the high cost of data collection limits scalability. Other approaches [3], [7], [8], [9] turn to data augmentation, self-supervised learning, and the incorporation of task priors to reduce this dependency. Meanwhile, large-scale pretraining on unlabeled data, like [10], [11], [12], [13], [14], has substantially improved transfer learning and enabled zero-shot capabilities. However, these approaches focus primarily on 2D representations, which may be suboptimal for robots operating in 3D spaces. To address this gap, we propose a framework for learning 3D visual representations: by learning geometric and semantic properties through latent point clouds, our method enhances robotic perception in 3D environments and improves performance in various downstream tasks.

B. 3D-aware Pretraining for robotics and embodied AI

The development of visual pretraining [3], [8], [9], [11], [10], [12] has shown an impressive generalization ability on robotic tasks. However, 2D pretraining approaches still suffer from the lack of spatial information. To address this issue, some works [15], [16], [17], [18], [19] typically attempt to lift 2D priors into 3D spaces and introduce spatial awareness into 2D foundation models. SPA [20] leverages neural volume rendering on multi-view images to endow a vanilla ViT with intrinsic 3D spatial awareness, but requires huge computing resources and training time to achieve the best performance. Lift3D [21] leverages 2D foundation models to guide the representation learning, but it still relies on the completeness of the point cloud observation.

C. Neural Rendering

Neural volume rendering-pioneered by NeRF [1]-provides a fully differentiable bridge between 2D views and 3D scenes, and has since been accelerated via proposal networks and factorization or by swapping in alternative implicit surfaces [22], [23], [14], [24], [25]. However, NeRF-based volumetric rendering methods often require sampling a large number of points along each ray, which incurs high computational cost and leads to slow convergence during training. More recently, 3D Gaussian Splatting (3DGS) [2] has emerged as an explicit, high-fidelity scene representation that splats learned Gaussian kernels into a volumetric field for ultra-fast rendering and incremental updates, and Feature Gaussian Splatting [26], [27], [28] further extends this by encoding rich point-wise features into each Gaussian-enabling not only real-time view synthesis but also efficient multi-view feature aggregation for downstream tasks.

III. Preliminaries

A. 3D-aware Visual Representation Learning

Visual pretraining has traditionally focused on learning representations from 2D images, where encoders are trained to capture semantic and appearance information useful for downstream tasks. While effective in purely visual domains, such representations often lack awareness of the 3D structure that is essential for embodied agents operating in physical environments. To inject 3D awareness into the learned features, recent works typically resort to differentiable 3D rendering pipelines, which enable self-supervised training by reconstructing views from multi-view images and camera poses. These pipelines provide the supervisory signal needed for encoders to acquire geometry-aware representations from purely 2D observations.

A representative paradigm is the NeRF-style framework. Let $\{I_i\}_{i=1}^{N}$ denote multi-view images and camera parameters $\{T_i\}_{i=1}^{N}$. A unified abstraction for NeRF-style pretraining can be expressed as

$$\hat{I}_k = \Psi_d\Big(R_{\text{ray}}\big(\Psi_e(I_{i=1}^{N}), T_k\big)\Big), \tag{1}$$

where Ψ_e is a multi-view feature extractor (e.g., a 3D CNN) that constructs a 3D feature volume, and R_{ray} denotes the NeRF renderer. The queried features are aggregated and decoded into the image domain by Ψ_d, and the synthesized image $\hat{I}_k$ is supervised against the ground truth I_k. The latent volume $\Psi_e(I_{i=1}^{N})$ then serves as the scene representation, which is compact and expressive. However, this representation remains entirely implicit, lacking explicit structural information that could be directly connected with downstream tasks. In addition, the ray-casting operation is computationally demanding.

Another paradigm builds upon the 3DGS pipeline. Feature maps are first extracted from each image and then projected

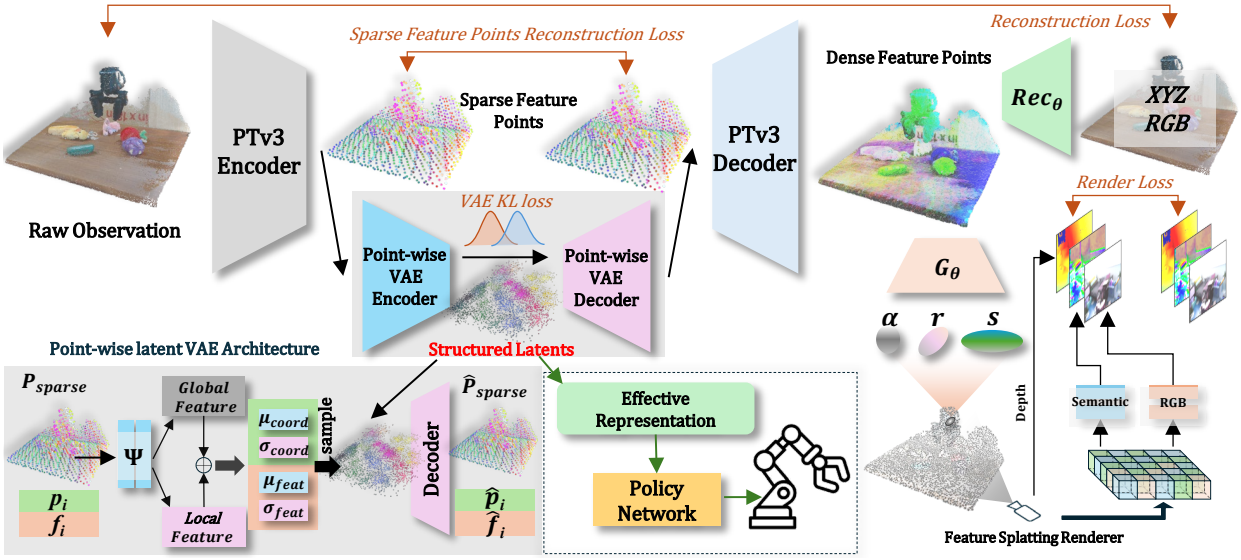

Fig. 1. The overview of our main pipeline.

into 3D space to form a feature point cloud. A point-based network predicts Gaussian splat parameters, and the scene is rendered via the GS renderer:

$$\hat{I}_k = R_{\text{rast}}\Big(\Phi_p\big(\pi_k * (\Phi_e(I_{i=1}^N), T_k)\big)\Big), \tag{2}$$

where Φ_e is a pretrained image encoder, π_k is the unprojection function under camera T_k, and Φ_p predicts the splat properties. Prior work has used either the predicted Gaussian parameters or the intermediate feature point set as the scene representation. R_{rast} is highly efficient. Nevertheless, the explicit and point-based nature inherently constrains them by resolution and limits their ability to capture compact, high-level abstractions of scene geometry.

Our framework IV-A incorporates a point-wise latent VAE into the latent space of a standard Point Transformer v3 (PTv3) to learn a structural latent representation, which is trained with a simple yet effective 3DGS-based self-supervised paradigm to endow the model with 3D awareness. The resulting representation combines the expressiveness of implicit methods with the structural cues of explicit ones, producing a compact, geometry-aware abstraction that is particularly well suited for downstream embodied tasks.

B. Gaussian Splatting

3DGS represents a 3D scene using a set of Gaussian primitives, each defined by a centroid $u \in \mathbb{R}^3$, and a set of properties related to its local shape, i.e. $\mathbf{G}_i = \{\Sigma_i = \mathbf{R}_{g,i}\mathbf{S}_{g,i}\mathbf{S}_{g,i}^T\mathbf{R}_{g,i}^T, o_i, c_i\}$ in which, to ensure stable backpropagation during optimization. The lower subscript i refers to the $i-th$ element in the Gaussian splat set. Σ_i is decomposed into a rotation matrix $\mathbf{R}_{g,i}$ and scale matrix $\mathbf{S}_{g,i}$. o_i, c_i denote the opacity and color of the $i-th$ splat. To preserve differentiability in the rendering process, a local affine approximation is applied by linearizing the projection using the Jacobian matrix J, i.e. $\Sigma' = JW\Sigma W^T J^T$ Then, the final color and depth map is defined via:

$$\mathbf{C} = \sum_{i\in\mathcal{N}} c_i\alpha_i\prod_{j=1}^{i-1}(1-\alpha_j), \quad \mathbf{D} = \sum_{i\in\mathcal{N}} d_i\alpha_i\prod_{j=1}^{i-1}(1-\alpha_j), \tag{3}$$

whered_i refers to the distance between $i-th$ splat and camera center. α_i denotes the soft occupation of the splat at the corresponding 2D location of the image plane, which can be obtained by $\alpha_i(x) = o_i exp(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^T(x-\mu_i))$.

IV. Methodology

A. Main Architecture of the Pretraining Framework

The overall architecture of our pretraining framework can be divided into two main components. The first component focuses on point cloud encoding and decoding for effective structured latents extraction, while the second component performs geometric reasoning and 2D feature rendering for multi-view supervision.

In the first stage, we adopt Point Transformer V3 (PTv3) together with a point-wise latent VAE. The raw point cloud $\mathbf{P}_{\text{raw}}$ is first encoded by the PTv3 encoder to obtain the sparse feature points, marked as: $\mathbf{P}_{\text{sparse}} = \text{PTv3}_{\text{enc}}(\mathbf{P}_{\text{raw}})$.

Typically, the sparse feature point cloud is directly fed into a PTv3 decoder that upsamples it to recover the original resolution, producing a dense feature point cloud. In contrast, the key distinction of our approach is that, before decoding, we insert a **point-wise latent VAE** (**PL-VAE**) into the encoders latent space. This additional reconstruction step is designed to enforce a compact and robust representation, ensuring stronger generalization for downstream tasks, which we will introduce in the next section in detail. The output of the **PL-VAE** is referred to as $\hat{\mathbf{P}}_{\text{sparse}}$ and passed to the decoder, which restores the point cloud resolution, stated as $\mathbf{P}_{\text{dense}} = \text{PTv3}_{\text{dec}}(\hat{\mathbf{P}}_{\text{sparse}})$.

The second component aims to equip the framework with 3D awareness through differentiable rendering.

This rendering component is deliberately kept simple, it includes only the common, effective, and lightweight modules inspired by prior 3DGS-based representation learning

[29], [30], [26]. Keeping the renderer simple ensures the first component remains the primary contributor of representation learning, preventing the second component from absorbing excessive burden or compensating for the encoder. In turn, this encourages the structural latent module to learn more informative and transferable features.

Concretely, the component comprises a geometric reasoning module G_θ, a feature-splatting module, and a reconstruction module Rec_θ. The module G_θ predicts point-wise Gaussian properties, as described in Sec. III-B, from the decoded dense feature points:

$$\mathbf{G} \;=\; G_\theta(\mathbf{P}_{\text{dense}})\,. \tag{4}$$

Reasoning over these spatial attributes enables 3DGS rendering, effectively bridging the PTv3 latent space with local geometry [29], [31], [25]. We then perform *feature splatting*, which extends conventional Gaussian rasterization to render point-wise feature information (and depth) rather than color. Given camera v, we splat features and depths only, stated in Eq. 5, rather than both features and color.

$$\left(\hat{\mathbf{F}}_v, \hat{\mathbf{D}}_v\right) \;=\; R_{\text{v}}(\mathbf{G}, \mathbf{P}_{\text{dense}})\,. \tag{5}$$

Prior works vary in this choice, but ablations consistently show that representation quality hinges on feature splatting, while color splatting mainly improves image fidelity [27], [26]. Since our goal is to learn geometry-aware features (not photorealistic images), feature-only splatting keeps the pipeline concise and focuses model capacity on representation learning. The splatted feature maps are then decoded by two projectors into semantic features and color:

$$\hat{\mathbf{I}}_v, \hat{\mathbf{F}}_v^s = H_c(\hat{\mathbf{F}}_v), H_s(\hat{\mathbf{F}}_v). \tag{6}$$

Unlike methods that render feature and color separately and align them post hoc, we apply both supervisions directly to a shared latent feature space, which simplifies the pipeline and tightly couples semantic and appearance cues, yielding stronger representations. We also align rendered and ground-truth depths, providing direct gradients to G_θ.

In addition to rendering alignment, we introduce a lightweight reconstruction module that regresses original color and 3D coordinates of each point from the decoded dense features. We add this module because, early in training, the rendered features are underdeveloped and provide relatively indirect supervision. The auxiliary reconstruction loss anchors the representation to basic appearance and geometry, accelerating convergence, stabilizing the initial training phase, and steering the features toward a well-behaved latent space for more robust learning overall.

B. Point-wise Latent VAE

Unlike traditional Variational Auto Encoder (VAE), which only produces a global feature for the point cloud, the PL-VAE operates both at the feature and coordinates of the latent sparse feature points ($\mathbf{P}_{sparse}$), constructing a 3D-aware structured latent.

For the encoder of the PL-VAE, we extract global and local descriptors from these sparse points:

$$f_{global}, f_i = \mathbf{Agg}(\Psi(\mathbf{P}_{sparse})), \mathbf{MLP}(\Psi(\mathbf{P}_{sparse})), \tag{7}$$

where **Agg** denotes the aggregation based on attention pooling. Ψ is a PointNet backbone for extracting features. The global features provide a holistic overview of the scene, while local features capture point-wise context.

For each point, we define a joint representation $h_i = cat(f_{global}, f_i)$ by concatenating the f_{global} and f_i. This joint representation h_i is then used to predict the Gaussian distribution parameters in the VAE, covering both the feature dimensions and the coordinate dimensions for each point.

$$(\mu_i^f, \sigma_i^f), (\mu_i^p, \sigma_i^p) = \phi_f(h_i), \phi_p(h_i), \tag{8}$$

where $\phi_f(\cdot)$ and $\phi_p(\cdot)$ are MLPs that output the mean and standard deviation for the respective distributions. We then sample latent variables using the reparameterization trick to form the intermediate features, that are regularized to lie close to a Gaussian distribution in the VAE latent space, referred to as: $\mathbf{z}_{vae} = (z_i^f, z_i^p)_{i=1}^N$. Compared with the $\mathbf{P}_{sparse}$, the $\mathbf{z}_{vae}$ are more compact and closer to the Gaussian prior. The point positions no longer represent precise geometry; instead, they behave like Gaussian-smoothed approximations. Such approximate structural encoding is advantageous in the latent space: it reduces sensitivity to noise and fine-grained variations while preserving the overall geometric tendencies, leading to smoother manifolds and stronger generalization for downstream tasks. Then, $\mathbf{z}_{vae}$ is decoded via a VAE decoder to reconstruct $\hat{\mathbf{P}}_{sparse}$, enabling cascaded connections within the overall framework and facilitating end-to-end joint optimization: $\hat{\mathbf{P}}_{sparse} = \Psi_{\text{dec}}(\mathbf{z}_{vae})$. After that, the reconstructed $\hat{\mathbf{P}}_{sparse}$ is fed to the PTv3 decoder.

C. Effective Representation for Downstream Task

The reparameterized latent representation $\mathbf{z}_{vae}$, obtained from the PL-VAE as described in Section IV-B, serves as the effective input for downstream embodied tasks. It provides a compact and smooth latent space regularized toward a Gaussian prior, capturing high-level structural regularities. This abstraction enhances robustness to noise and scene variations, improves generalization across domains.

In downstream applications, $\mathbf{z}_{vae}$ can be directly used as the input to a policy network, bypassing the need for explicit geometric reconstruction. For example, in robot imitation learning, the policy network π receives the latent point cloud representation $\mathbf{z}_{vae}$ along with the robot's proprioceptive state O_{agent} (such as joint positions, velocities, or gripper status), and outputs a corresponding action $\hat{a}$:

$$\hat{a} = \pi(\mathbf{z}_{vae}, O_{agent}). \tag{9}$$

D. Optimization Strategies

This section describes the optimization strategies and loss functions employed in this framework.

According to Sec. IV-A and IV-B, the fundamental loss is the rendering alignment of the decoded RGB and semantic

feature map, as well as the rendered depth map (Eq. 5 and 6). This can be described as:

$$L_{render} = \frac{1}{V}\sum_{i=1}^{V}\Big(\beta_1 \cdot \|\mathbf{I}_v - \hat{\mathbf{I}}_v\|_1 + \beta_2 \cdot \|\mathbf{D}_v - \hat{\mathbf{D}}_v\|_1 + \beta_3 \cdot \|\mathbf{F}_v^s - \hat{\mathbf{F}}_v^s\|_1\Big). \tag{10}$$

The symbols here retain the same meaning as the above text. The groundtruth of semantic maps $\mathbf{F}_s^v$ is produced by E-RADIO [32]. In our experiments, we set $\beta_1 = 1, \beta_2 = 0.2, \beta_3 = 0.1$ constantly.

Moreover, we introduce an auxiliary reconstruction loss to map the $\mathbf{P}_{dense}$ to the original $\mathbf{P}_{raw}$, thereby improving both convergence speed and training stability.

$$L_{recon} = \frac{1}{N}\sum_{i=1}^{n}(\|p_i - \hat{p}_i\|_1 + \|c_i - \hat{c}_i\|_1). \tag{11}$$

The N denotes the number of points, p_i, c_i represent the raw point coordinate and color.

Additionally, we introduce two loss terms to supervise the PL-VAE depicted. The first term is the L1 latent reconstruction loss, aiming to reconstruct the $\mathbf{P}_{sparse}$ with VAE:

$$L_{vae} = \frac{1}{M}\sum_{m}^{1}(\omega_1\|p_i - \hat{p}_i\| + \omega_2\|f_i - \hat{f}_i\|), \tag{12}$$

in which p_i, f_i are the coordinates and features of the $\mathbf{P}_{sparse}$. We use ω_1 and ω_2 to balance the reconstruction of point positions and features, since, as shown in Eq. 8, they are sampled from Gaussian parameters predicted by different heads and thus exhibit different statistical scales. In our experiments, the $\omega_1 = 1, \omega_2 = 0.1$ Conventional VAEs for point generation often employ the Chamfer Distance loss to align the overall distribution of points. However, their objective is typically generation rather than exact recovery. In our case, the goal is to precisely reconstruct $\mathbf{P}_{sparse}$, and therefore we adopt an L1 loss instead.

Furthermore, the classic VAE KL loss is introduced as well:

$$L_{KL} = \mathrm{KL}(\mathcal{N}(\mu_i^f, \sigma_i^f)\,\|\,\mathcal{N}) + \mathrm{KL}\big(\mathcal{N}(\mu_i^p, \sigma_i^p)\,\|\,\mathcal{N}\big). \tag{13}$$

This term encourages both the feature- and coordinate-level posteriors to remain close to a standard Gaussian prior $\mathcal{N}$, promoting a smooth and well-behaved latent space. Therefore, the total loss terms can be described as:

$$L_{total} = (1 - w(t))L_{render} + w(t)L_{recon} + L_{vae} + L_{KL}. \tag{14}$$

Here $w(t)$ is an annealing weight that starts at 1 and gradually decays to 0.1. This scheduling emphasizes reconstruction early, when rendered features are immature, and progressively shifts focus to rendering alignment as training stabilizes.

V. Experiments

Benchmarks. In this section, we present overall evaluations of the proposed pretraining methods. During recent years, the development of robotic simulators has significantly enhanced for reproducible and efficient robot performance evaluation. Hence, we selected two widely-adopted simulator benchmarks: RLBench [33], and Maniskill2 [34]. The two benchmarks are chosen to be diverse and representative. Furthermore, we establish a real-world robot platform to validate the effectiveness of our method on real-world cases. Our goal is to evaluate the performance of the pretrained representation; therefore, we uniformly pretrain all baselines on the same dataset ScannetV2 before we adopt them for the downstream manipulation tasks.

Metrics. We use the Success Rate (S.R.) to assess the performance of each method on individual tasks. Additionally, following the evaluation protocol from VC-1 [35] and PCM [36], we report two aggregate metrics: Mean S.R. and Mean Rank. Mean S.R. reflects the average success rate across all tasks. While mean Rank ranks methods by their success rate on each task and then averages these ranks across all tasks.

A. Evaluation on Simulation benchmarks

We evaluate our approach on RLBench and ManiSkill2 using OBSBench [36], which provides standardized pipelines and a diverse set of baselines within a unified framework for benchmarking robotic performance. Following the OBSBench protocol, we adopt its default settings and train an Action Chunking Transformer (ACT) policy [37]. For comparison, we include three backbone models: MultiViT [7], PTv3 [38], and SPUNet [39] trained from scratch under an autoencoder paradigm, as well as several pretrained 3D-aware approaches, including VC-1 [35], MultiMAE [40], PonderV2 [41], GSRL [30], [42], SPA [20], and Lift3D [21].

RLBench. Table I reports S.R. across nine RLBench tasks. We observe that 3D-aware pretrained methods consistently outperform backbone models trained from scratch, highlighting the importance of visual representation learning. Furthermore, point-based pretraining paradigms generally yield stronger results than multi-view image pipelines. Our method achieves the best overall performance, which we attribute to its integration of implicit latents with explicit structural cues.

Maniskill2. For the Maniskill2 platform, we selected 3 rigid body tasks (PickCube, StackCube, TurnFaucet) and 3 soft body tasks (Hang, Fill, and Pour) for evaluation. The full results can be seen in Table II. Across all six ManiSkill2 tasks, our method consistently delivers the most substantial and stable performance over existing pretrained baselines. This underscores the superior generalizability of our pretraining approach.

B. Further Analysis and Ablation Studies

In this section, we explain a more in-depth analysis of the proposed method and ablate all necessary components.

Ablation of the geometric reasoning module. To validate the significance of local geometric information, we remove the geometric reasoning module and assign some constant value to the properties of all Gaussian points. Specifically, we fix their opacity as 1, scale as (0.001, 0.001, 0.001), and rotation as (1, 0, 0, 0). Then we directly adopt these assigned

TABLE I

RESULTS ON RLBENCH TASKS UNDER ACT POLICY

Task	MultiViT	PTV3	SpUNet	VC-1	MultiMAE	PonderV2	GSRL	Lift3D	No.VAE	Ours
OpenDrawer	0.08	0.40	0.44	0.24	0.36	0.60	0.52	0.54	0.56	**0.62**
SweepTo	**1.00**	0.96	0.90	0.96	**1.00**	0.96	0.92	**1.00**	**1.00**	**1.00**
MeatoffGrill	0.36	0.68	0.72	0.12	0.04	0.72	0.60	0.74	0.76	**0.80**
TurnTap	0.00	0.04	0.00	0.04	0.08	0.00	0.02	0.04	0.08	**0.12**
ReachandDrag	0.04	0.04	0.20	0.20	0.16	**0.28**	0.26	0.16	0.24	**0.28**
PutMoney	0.28	0.44	0.60	0.72	0.56	0.64	0.62	0.58	0.48	**0.76**
PushButtons	0.14	0.64	0.00	0.44	0.48	0.16	0.40	0.72	0.68	**0.74**
CloseJar	0.00	0.44	0.04	0.08	0.00	0.28	0.04	**0.54**	0.48	0.52
PlaceWine	0.00	0.04	0.00	0.00	0.08	0.16	0.08	**0.20**	0.16	0.18
Mean S.R.↑	0.21	0.41	0.32	0.31	0.31	0.42	0.38	0.50	0.49	**0.56**
Mean Rank↓	8.56	6.50	7.39	6.50	6.44	4.83	6.11	3.61	3.56	**1.50**

TABLE II

RESULTS ON MANISKILL2 TASKS UNDER ACT POLICY

Task	MultiViT	PTV3	SpUNet	VC-1	MultiMAE	SPA	PonderV2	GSRL	Lift3D	No.VAE	Ours
PickCube	0.04	0.76	0.74	0.77	0.52	0.62	0.87	0.76	0.90	0.82	**0.92**
StackCube	0.00	0.08	0.22	0.06	0.30	0.00	0.35	0.28	0.35	0.32	**0.38**
TurnFaucet	0.35	0.00	0.39	0.42	0.37	0.45	0.30	0.28	0.45	0.44	**0.48**
Hang	0.84	0.88	0.84	0.84	0.77	0.92	0.83	0.88	**1.00**	0.96	0.98
Pour	0.00	0.08	0.10	0.04	0.00	0.00	0.11	0.12	0.15	0.10	**0.20**
Fill	0.76	0.56	0.66	0.78	0.68	0.86	0.73	0.70	**0.90**	0.88	0.89
Mean S.R.↑	0.33	0.39	0.49	0.49	0.44	0.48	0.53	0.50	0.63	0.59	**0.64**
Mean Rank↓	8.92	8.17	7.42	6.67	8.67	6.67	5.92	6.50	1.83	3.92	**1.33**

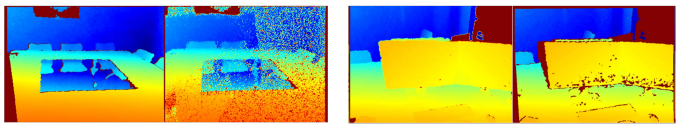

Fig. 2. Comparison of the rendered depth map with or without reasoned geometric properties.

TABLE III

ABLATION STUDIES OF THE GEOMETRIC REASONING MODULE.

	RLBench			Maniskill	
	TurnTap	CloseJar	MeatOffGrill	StackCube	Hang
No Reasoned Geo.	0	0.24	0.72	0.28	0.92
Ours	**0.08**	**0.48**	**0.76**	**0.36**	**0.96**

properties to splat features. Fig. 2(a) gives an example of the comparison of the rendered depth with or without the geometric reasoning module. The depth map rendered by reasoned geometry has a smooth and accurate depth map generated by reasoned geometry, while the right depth map using a constant property-results in noisy and inaccurate geometry, especially around object boundaries. This comparison highlights the importance of geometry reasoning for generating clean and reliable depth information.

Fig. 2(b) illustrates that even compared with the groundtruth captured by Lidar, which sometimes misses points on object surfaces, our method produces a more complete and cleaner depth map. This demonstrates the geometry reasoning ability of our approach. Additionally, we evaluate the pretrained model of the two setups by the downstream embodied tasks, and the results are listed in Table III, which shows that incorporating reasoned geometry significantly improves task performance across both RLBench and ManiSkill2 benchmarks. Compared to the variant without geometry reasoning, our method achieves consistently higher success rates, with notable gains on challenging tasks like CloseJar.

Ablation of the Point-wise latent VAE module. One of the most impactful components of our approach is the PL-VAE (point-wise latent VAE) applied in the PTv3 encoders latent space, enabling joint optimization of point-wise coordinates and features. We ablate this design by removing the PL-VAE and evaluating on all simulated datasets (RLBench and ManiSkill2); the results are included in Tables I and II. As shown, eliminating the PL-VAE leads to substantial drops across all tasks. Although the model without PL-VAE still outperforms a PTv3-from-scratch baseline, it offers only a limited advantage over other 3D-aware methods. In contrast, reinstating the PL-VAE yields large, consistent gains and achieves the best overall performance, corroborating its role in learning compact, well-regularized structural latents.

Ablation of the feature splatting. As we discussed in Sec. IV-A, typical methods splat both feature and color simultaneously, and build their respective loss terms, while we investigate and found that only feature terms provide the most contributions for representation learning. Here we provide a qualitative example in Fig. 3 to illustrate that. From this figure, we can observe that the features predicted by our solution exhibit clearer semantic separability, making distinct objects and boundaries more easily distinguishable.

Complexity Analysis. We compare the training efficiency of our method with VC-1, SPA, and PonderV2, including the training GPU time and required device. Relative to prior 3D-aware baselines, our setup reduces aggregate GPU time by 4.6, 24, and 60 times, respectively. Just as importantly, it removes the need for large distributed clusters, simplifying reproduction and hyperparameter tuning by avoiding cross-GPU communication and schedule engineering. This smaller

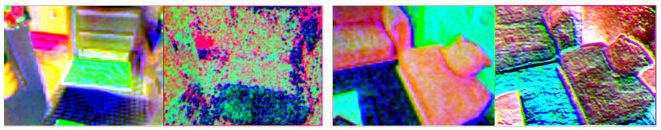

Fig. 3. PCA visualizations of extracted features. The post feature decoding denotes the results of our feature-only rendering.

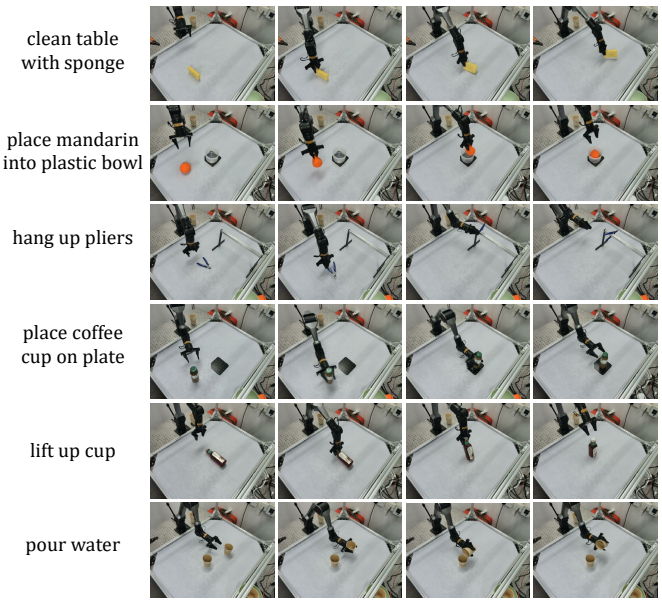

Fig. 4. Real-world task demonstration of the proposed approach.

hardware footprint translates to lower operational cost and energy use. This advantage becomes even more pronounced when scaling up to larger datasets or training large models such as world models. Therefore, our method offers better scalability for large-scale pretraining.

TABLE IV

COMPARISON OF TRAINING SETUPS FOR DIFFERENT METHODS.

	VC-1	SPA	PonderV2	Ours
Device	-	80×A100	8×A100	1×A800
Epoch	-	2000	100	400
GPU Time	∼10000h	∼4000h	∼768h	∼166h

C. Evaluation on Real-world Experiments

We deployed our pretrained representation encoder on a real-world robotic platform to further evaluate its generalization capabilities. Specifically, we utilized an Orbbec Femto Bolt time-of-flight camera to capture RGBD information, which is fed to the PTv3 encoder, Eq 7, and 8 to obtain an overall representation of the observation. The robotic system was equipped with an AgileX Piper robotic arm for policy execution. For the control policy, we employed ACT [37]. Our experiments were conducted across six distinct manipulation tasks as shown in Fig. 4: (1) clean table with sponge, (2) place toy pea into plastic bowl, (3) hang up pliers, (4) place coffee cup on plate, (5) lift up cup, and (6) pour water. For each task, we collected expert demonstrations consisting of an action sequence spanning 360 steps at 30 Hz.

TABLE V

SUCCESS RATE ON THREE TASKS ACROSS FOUR METHODS

	Pointnet	PonderV2	Ours (scratch)	Ours
place mandarin	56	72	76	**84**
lift up bottle	64	48	40	**68**
place starbucks	36	32	28	**36**

The dataset was split into a training set of 40 demonstrations and a test set of 10 demonstrations. Training was performed on a single NVIDIA RTX3090 GPU for 10000 epochs using the AdamW optimizer with a fixed learning rate of $5e-5$. The successful deployment of the trained model in real-world manipulation tasks underscores the practicality of our approach. Table V reports the success rate of some real robot tasks, where Ours (scratch) denotes our model without the pretraining. The results show that our model consistently outperforms the other conterparts.

VI. DISCUSSION

Conclusion. This paper introduces a 3D pretraining framework that fuses the strengths of implicit and explicit scene representations. By introducing a PL-VAE in the PTv3 latent space and regularizing both point features and coordinates, the method learns structural latent points that are compact, smooth, and geometry-aware. A streamlined 3DGS setup with predicted Gaussian properties, feature-only splatting, and depth alignment keeps supervision focused on representation quality, while an auxiliary point/color reconstruction stabilizes early training. Across RLBench and ManiSkill2, the approach achieves the best overall results among the compared 3D-aware methods and remains robust across tasks; ablations confirm that both the geometric reasoning module and the PL-VAE are key contributors. Finally, real-robot evaluations further support the methods transferability beyond simulation.

Limitation. Since the method introduces a sampling step, it improves generalization ability but cannot represent object geometry with complete precision. Therefore, in scenarios that demand extremely high accuracy, such as surgical operations or threading a needle, its upper bound performance may be limited. Future research could explore 3D representations that couple deterministic and probabilistic components.

ACKNOWLEDGEMENT

This work is partially supported by the National Natural Science Foundation of China (No. 62306261), HK RGC-Early Career Scheme (No. 24211525), ITSP Platform Project (No. ITS/600/24FP) and the SHIAE Grant (No. 8115074). This study was supported in part by the Centre for Perceptual and Interactive Intelligence, a CUHK-led InnoCentre under the Hong Kong SAR Governments InnoHK initiative. This is also partially supported by Hong Kong RGC Strategic Topics Grant (No. STG1/E-403/24-N), and CUHK-CUHK(SZ)-GDST Joint Collaboration Fund (No. YSP26-4760949).

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis *et al.*, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[3] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4788–4795.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[5] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[6] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik, "Robot learning with sensorimotor pre-training," in *Conference on Robot Learning*. PMLR, 2023, pp. 683–693.

[7] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International conference on learning representations*, 2021.

[8] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19 884–19 895, 2020.

[9] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter *et al.*, "Scaling robot learning with semantically imagined experience," *arXiv preprint arXiv:2302.11550*, 2023.

[10] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[12] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[13] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 5639–5650.

[14] D. Huang, S. Peng, T. He, H. Yang, X. Zhou, and W. Ouyang, "Ponder: Point cloud pre-training via neural rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 16 089–16 098.

[15] P. Wang, Z. Fan, Z. Wang, H. Su, R. Ramamoorthi *et al.*, "Lift3d: Zero-shot lifting of any 2d vision model to 3d," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 367–21 377.

[16] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," *arXiv preprint arXiv:2402.10885*, 2024.

[17] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox, "Rvt-2: Learning precise manipulation from few demonstrations," *arXiv preprint arXiv:2406.08545*, 2024.

[18] D. Niu, Y. Sharma, H. Xue, G. Biamby, J. Zhang, Z. Ji, T. Darrell, and R. Herzig, "Pre-training auto-regressive robotic models with 4d representations," *arXiv preprint arXiv:2502.13142*, 2025.

[19] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation," *arXiv preprint arXiv:2306.17817*, 2023.

[20] H. Zhu, H. Yang, Y. Wang, J. Yang, L. Wang, and T. He, "Spa: 3d spatial-awareness enables effective embodied representation," *arXiv preprint arXiv:2410.08208*, 2024.

[21] Y. Jia, J. Liu, S. Chen, C. Gu, Z. Wang, L. Luo, L. Lee, P. Wang, Z. Wang, R. Zhang *et al.*, "Lift3d foundation policy: Lifting 2d large-scale pretrained models for robust 3d robotic manipulation," *arXiv preprint arXiv:2411.18623*, 2024.

[22] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," *arXiv preprint arXiv:2204.02296*, 2022.

[23] J. Wang, T. Bleja, and L. Agapito, "Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction," in *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022, pp. 433–442.

[24] J. Wang, Z. Zhang, and R. Xu, "Learning robust generalizable radiance field with visibility and feature augmented point representation," *arXiv preprint arXiv:2401.14354*, 2024.

[25] J. Wang, J. He, Z. Zhang, M. Sun, J. Sun, and R. Xu, "Evggs: A collaborative learning framework for event-based generalizable gaussian splatting," *arXiv preprint arXiv:2405.14959*, 2024.

[26] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, "Feature splatting: Language-driven physics-based scene synthesis and editing," *arXiv preprint arXiv:2404.01223*, 2024.

[27] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi, "Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 676–21 685.

[28] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, "Langsplat: 3d language gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 051–20 060.

[29] J. Wang, Z. Zhang, J. He, and R. Xu, "Pfgs: High fidelity point cloud rendering via feature splatting," in *European Conference on Computer Vision*. Springer, 2024, pp. 193–209.

[30] J. Wang, Q. Zhang, J. Sun, J. Cao, G. Han, W. Zhao, W. Zhang, Y. Shao, Y. Guo, and R. Xu, "Reinforcement learning with generalizable gaussian splatting," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[31] B. Zhou, S. Zheng, H. Tu, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. Liu, "Gps-gaussian+: Generalizable pixel-wise 3d gaussian splatting for real-time human-scene rendering from sparse views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[32] M. Ranzinger, G. Heinrich, J. Kautz, and P. Molchanov, "Am-radio: Agglomerative vision foundation model – reduce all domains into one," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 12 490–12 500.

[33] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[34] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," *arXiv preprint arXiv:2302.04659*, 2023.

[35] A. Majumdar, K. Yadav, S. Arnaud, J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. Vakil *et al.*, "Where are we in the search for an artificial visual cortex for embodied intelligence?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 655–677, 2023.

[36] H. Zhu, Y. Wang, D. Huang, W. Ye, W. Ouyang, and T. He, "Point cloud matters: Rethinking the impact of different observation spaces on robot learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 77 799–77 830, 2024.

[37] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[38] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler faster stronger," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4840–4851.

[39] X. Liu, X. Liu, Y.-S. Liu, and Z. Han, "Spu-net: Self-supervised point cloud upsampling by coarse-to-fine reconstruction with self-projection optimization," *IEEE Transactions on Image Processing*, vol. 31, pp. 4213–4226, 2022.

[40] R. Bachmann, D. Mizrahi, A. Atanov, and A. Zamir, "Multimae: Multi-modal multi-task masked autoencoders," in *European Conference on Computer Vision*. Springer, 2022, pp. 348–367.

[41] H. Zhu, H. Yang, X. Wu, D. Huang, S. Zhang, X. He, H. Zhao, C. Shen, Y. Qiao, T. He *et al.*, "Ponderv2: Pave the way for 3d foundation model with a universal pre-training paradigm," *arXiv preprint arXiv:2310.08586*, 2023.

[42] J. Wang, Z. Zhang, Q. Zhang, J. Li, J. Sun, M. Sun, J. He, and R. Xu, "Query-based semantic gaussian field for scene representation in reinforcement learning," *arXiv preprint arXiv:2406.02370*, 2024.