Appendix

This appendix provides additional details on implementation, evaluation metrics, and qualitative results that could not be included in the main document due to space constraints. The document is organized as follows:

- **Section A** presents extended experiments in which Comp3D-$\mathcal{G}$ is trained on multiple diverse datasets to improve its scalability and generalizability.
- **Section B** provides additional implementation details regarding the architecture of the Gaussian Decoder and the training details for Any-Feature 3D Lifting.
- **Section C** details the experimental settings, including dataset descriptions, baseline configurations, and evaluation protocols for efficient novel view synthesis, 3D scene understanding, and multi-view feature encoding.
- **Section D** presents additional ablation studies comparing compact reconstruction strategies and analyzing components of Comp3D-$\mathcal{F}$, followed by discussions on the autoencoder design, FPS analysis, and limitations.
- **Section E** provides additional quantitative experimental results, including latent decoding or two-view novel view synthesis.
- **Section F** presents additional qualitative results, including visualizations of attention maps, 3D scene understanding, feature PCA, and renderings from 24-view inputs.

A. Multi Dataset Training

We extend Comp3D-$\mathcal{G}$ to improve its scalability and generalizability by incorporating diverse training datasets, increasing the number of input views, and scaling up the number of Gaussians. However, since each dataset exhibits different camera parameters and scene scales, naively combining them makes training C3G unstable and difficult to converge. To address this, we first describe the architectural modifications made to accommodate these variations (§ A.1), followed by implementation details covering dataset configurations and training procedures (§ A.2). Finally, we present experimental results demonstrating the effectiveness of this extension (§ A.3).

A.1. Methodology

Architecture modification. To support training on diverse datasets with varying numbers of input views and an increased number of Gaussians, we introduce minimal modifications to the original Comp3D-$\mathcal{G}$ architecture. To fully exploit the pretrained weights of Comp3D-$\mathcal{G}$, which were already well-trained on the RealEstate10K dataset [52], we keep the architectural changes as minimal as possible to preserve the original structure.

First, to handle differences in the intrinsics of the camera and the scale of the scene between training datasets, we incorporate the intrinsic embeddings as follows [19, 48]. Specifically, we additionally take camera intrinsics as input and pass them through a projection layer consisting of a 2-layer MLP initialized with zero weights. The projected intrinsic features are then added to the patch-embedded features within VGGT [42].

Next, we increase the number of Gaussians to better cover larger scenes and capture finer details. To achieve this, we replicate the weights of the Gaussian head. In the original architecture, the Gaussian head generates one Gaussian per token. By replicating its weights $N_{\mathbf{G}}$ times, we instead produce $N_{\mathbf{G}}$ Gaussians per token, resulting in a total of $N \times N_{\mathbf{G}}$ Gaussians. Since the replicated heads share the same initialization, Gaussians originating from the same token are localized in the same spatial region, encouraging them to focus on capturing finer local details.

Additional loss. To improve training stability, we introduce additional geometry supervision in the form of depth and normal losses, encouraging the network to develop a stronger understanding of scene geometry. We first extract pseudo ground-truth normal maps N_t and depth maps D_t from the point maps produced by VGGT [42] for the corresponding view I_t. We then render the depth maps $\hat{D}_t$ and normal maps $\hat{N}_t$ using the same rasterizer as 3DGS [20], replacing the color attributes with z-value and surface normals. For supervision, we apply scale-shift invariant loss term for depth and MSE loss for normals,

$$\mathcal{L}_{\text{depth}} = \|(\alpha \cdot D_t + \beta) - \hat{D}_t\|, \tag{1}$$

$$\mathcal{L}_{\text{normal}} = \|N_t - \hat{N}_t\|, \tag{2}$$

where α and β is the scale and shift parameters obtained via least-squares alignment between rendered depth and pseudo ground-truth depth, following [43].

The overall loss term is followed as:

$$\mathcal{L} = \mathcal{L}_{\text{novel}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}}, \tag{3}$$

where λ_{depth} and λ_{normal} denotes the loss weight of normal and depth.

A.2. Implementation details

For efficient training, we initialize from the pretrained weights of Comp3D-$\mathcal{G}$ which were trained solely on the RealEstate10K dataset [52]. We set the number of Gaussians per query token to $N_{\mathbf{G}} = 16$. During training, we randomly sample 2 to 24 images per scene, with a maximum of 192 images per epoch, and train at a resolution of 224×224. We use the AdamW optimizer [21] with a learning rate of 1e-6. The geometry loss weights are set to $\lambda_{\text{depth}} = 0.01$ and $\lambda_{\text{normal}} = 0.001$. We train for 150K steps in total, with all other hyperparameters following Comp3D-$\mathcal{G}$.

For training, we incorporate 8 datasets spanning diverse domains. Specifically, we use RealEstate10K [52],

Table 10. **Comparison of novel view synthesis with multi-view input images on DL3DV [27].** Our method generates fewer Gaussians while achieving competitive or superior quality.

Methods	12 view				24 view				36 view			
	PSNR↑	SSIM↑	LPIPS↓	#G↓	PSNR↑	SSIM↑	LPIPS↓	#G↓	PSNR↑	SSIM↑	LPIPS↓	#G↓
AnySplat [19]	17.004	0.439	0.431	570K	17.263	0.445	0.439	1,140K	18.286	0.525	0.286	1,576K
VGGT+NoPo [42, 48]	11.714	0.320	0.550	602K	10.285	0.262	0.607	1,204K	9.65	0.236	0.628	1,806K
C3G (Ours)	18.768	0.505	0.408	2K	18.843	0.511	0.404	2K	18.675	0.504	0.412	2K
C3G++ (Ours)	19.368	0.512	0.431	32K	19.570	0.521	0.425	32K	19.736	0.523	0.424	32K

DL3DV [27], HyperSim [36], WildRGBD [46], TartanAir [44], ARKitScenes [3], BlendedMVS [47], and MapFree [1]. These datasets collectively cover a wide range of scenarios, including indoor, outdoor, and object-centric scenes across both real and synthetic domains.

A.3. Experiments

We evaluate novel-view synthesis performance on the test split of the DL3DV dataset [27]. As shown in Tab. 10, our model outperforms the baselines, benefiting from its improved generalizability and the increased number of Gaussians, which together enable better coverage of large scenes and finer detail capture. This is particularly evident on DL3DV, which predominantly consists of outdoor scenes with large spatial extents that are difficult to represent with only 2K Gaussians. By scaling up the number of Gaussians, our model can cover a significantly larger portion of each scene, leading to improved reconstruction quality. Furthermore, the results demonstrate that our model is robust to varying numbers of input views.

B. Additional implementation details

B.1. Details of Gaussian decoder

Architecture details. We first extract feature maps $\mathbf{F}_v = \mathcal{E}(I_v)$ from the visual encoder $\mathcal{E}(\cdot)$ given V input images $\{I_v\}_{v=1}^V$. For VGGT [42] which supports multi-view inputs, we directly obtain V feature maps $\{\mathbf{F}_v\}_{v=1}^V = \mathcal{E}(\{I_v\}_{v=1}^V)$ in a single feed-forward pass. For DINOv3 [39], we obtain V feature maps by passing each image to the encoder independently. We then concatenate the feature maps with the learnable tokens $\mathbf{Q}$, which are randomly initialized. This concatenated representation is processed through L transformer layers $\mathcal{T}_\mathcal{G}(\cdot)$, which consists of self-attention layers and MLP layers with ReLU activation functions and layer normalization [2]:

$$[\bar{\mathbf{Q}}; \bar{\mathbf{F}}] = \mathcal{T}_\mathcal{G}([\mathbf{Q}; \mathbf{F}]), \tag{4}$$

where $[\cdot, \cdot]$ denotes concatenation of dimension axis, $\bar{\mathbf{Q}}$ denotes refined learnable tokens and $\bar{\mathbf{F}}$ denotes refined features. In the self-attention layer, learnable tokens and the feature tokens are each projected to query, key, and value features to process the attention calculation. The query, key and value features of learnable tokens $\mathbf{Q}$ and visual encoder features $\mathbf{F}$ is calculated as follows:

$$Q = [Q_G; Q_I] = \mathcal{P}_Q([\mathbf{Q}; \mathbf{F}]), \tag{5}$$
$$K = [K_G; K_I] = \mathcal{P}_K([\mathbf{Q}; \mathbf{F}]), \tag{6}$$
$$V = [V_G; V_I] = \mathcal{P}_V([\mathbf{Q}; \mathbf{F}]), \tag{7}$$

where Q_G, Q_I denotes query features, K_G, K_I denotes key features, and V_G, V_I denotes value features of learnable token and visual encoder features, respectively. And $\mathcal{P}_Q$, $\mathcal{P}_K$, and $\mathcal{P}_V$ denote the projection layer of query, key, and value, respectively.

Then, the resulting output of self-attention layer in $\mathcal{T}_G$ is computed as:

$$\mathsf{Attn}(Q, K, V) = \mathsf{Softmax}(\frac{QK^\intercal}{\sqrt{d_Q}})V, \tag{8}$$

where d_Q denotes the channel dimension of query features.

Each of the refined learnable tokens $\bar{\mathbf{Q}}_i$ are decoded as a single Gaussian $\mathbf{G}_i$ by passing the Gaussian head $H_\mathcal{G}(\cdot)$, which consists of a single linear layer:

$$\mathbf{G}_i = H_\mathcal{G}(\bar{\mathbf{Q}}_i). \tag{9}$$

Attention visualization. To visualize the attention map in Fig. 3-(b), we select one Gaussian $\mathbf{G}_i$ and its corresponding learnable tokens $\mathbf{Q}_i$. We then calculate the attention weights between the query features of the learnable token Q_{G_i} and the key features K_I as follows:

$$A_i = Q_{G_i} \cdot K_I^\intercal, A_i \in \mathrm{R}^{N_\text{head} \times h \times w}, \tag{10}$$

where N_head denotes number of heads. We average the A_i at the head dimension, then apply min-max normalization for better visualization.

B.2. Details of feature decoder

Architecture details. We initialize Comp3D-$\mathcal{F}$ by copying the Comp3D-$\mathcal{G}$'s architecture and parameters, except for the learnable tokens and the Gaussian head. We first extract feature maps $\mathbf{F}'_v = \mathcal{E}'(I_v)$ using the user-desired visual encoder $\mathcal{E}'(\cdot)$. We then introduce new learnable feature tokens $\mathbf{Q}'$ and concatenate them with $\mathbf{F}_v$, which are then processed through $\mathcal{T}_\mathcal{F}(\cdot)$ with the same architecture as $\mathcal{T}_\mathcal{G}(\cdot)$:

$$[\bar{\mathbf{Q}}'; \bar{\mathbf{F}}'] = \mathcal{T}_\mathcal{F}([\mathbf{Q}'; \mathbf{F}']), \tag{11}$$

where $\bar{\mathbf{Q}}'$ denotes the refined learnable tokens and $\bar{\mathbf{F}}'$ denotes the refined features. In the self-attention layers of $\mathcal{T}_{\mathcal{F}}$, we reuse query Q and key K features from $\mathcal{T}_{\mathcal{G}}$ with stop-gradients, while only the value projection layer is newly trained:

$$V' = [V'_G; V'_I] = \mathcal{P}'_V([\mathbf{Q}'; \mathbf{F}']), \tag{12}$$

where V'_G, V'_I denote the value features of the learnable feature tokens and features from $\mathcal{E}'$, respectively, and $\mathcal{P}'_V$ denotes the projection layer of value in $\mathcal{T}_{\mathcal{F}}$.

Then, the resulting output of self-attention layer in $\mathcal{T}_{\mathcal{F}}$ is computed as:

$$\mathsf{Attn}(Q, K, V') = \mathsf{Softmax}(\frac{QK^{\intercal}}{\sqrt{d_Q}})V'. \tag{13}$$

These refined learnable feature tokens $\bar{\mathbf{Q}}'$ are converted to multi-view aggregated features $\mathbf{F}''_i$ for each Gaussian $\mathbf{G}_i$ by passing the feature head $H_{\mathcal{F}}$, which consists of a single linear layer:

$$\mathbf{F}''_i = H_{\mathcal{F}}(\bar{\mathbf{Q}}'_i). \tag{14}$$

Note that the introduction of new learnable queries $\mathbf{Q}'$ is to enable Comp3D-$\mathcal{F}$ to take features extracted from any user-desired visual encoder $\mathcal{E}'(\cdot)$ as input, considering that different features $(\mathcal{E}(\cdot), \mathcal{E}'(\cdot))$ will have different feature dimensions. When the two visual encoders are identical $(\mathcal{E}(\cdot) = \mathcal{E}'(\cdot))$, we can additionally copy the learned queries $\mathbf{Q}$ from Comp3D-$\mathcal{G}$, only training the value projection layer $\mathcal{P}'_V(\cdot)$ and the feature head $H_{\mathcal{F}}(\cdot)$.

C. Experimental settings

C.1. Novel view synthesis

Datasets. To evaluate novel view synthesis, we use the RealEstate10K [52] dataset. We adopt the same train-test split as prior work [5, 6, 48]. RealEstate10K primarily contains indoor real estate videos with camera poses computed using COLMAP [38]. For multi-view evaluation, we randomly sample 1,000 image sets from the test split, each consisting of multiple context views and 3 target views that are disjoint from the context views.

Baselines. For 2-view input settings, we compare against SOTA generalizable feed-forward methods on novel view synthesis. We compare PixelSplat [5] and MVSplat [6] as *Pose-dependent* categories, which require ground-truth pose information for input. We also compare our model with *Pose-free* categories which use only rgb images as inputs such as CoPoNeRF [17], Splatt3R [40], PF3plat [16], SPFSplat [18], NoPoSplat [48]. We also reimplement the VGGT+NoPo, which replaces NoPoSplat's MASt3R [24] backbone with VGGT [42] while maintaining NoPoSplat's pipeline to estimate per-pixel Gaussians. Ours also fall into *Pose-free* settings.

For the multi-view ($V > 2$) settings, we compare against state-of-the-art generalizable methods that support arbitrary numbers of input views, such as AnySplat [19]. We evaluate using various numbers of input views: 12, 24, and 36. We also evaluate VGGT+NoPo under the same settings. For AnySplat, we provide input images at 448×448 resolutions, which perform better than 256×256, and render target view images at 256×256 resolution.

Evaluation protocol. Given unposed images as inputs, our method reconstructs and represents 3D scenes using 3D Gaussians. We rasterize the 3D Gaussians at ground-truth camera poses and compare the rendered images with ground-truth target views. We report standard novel view synthesis metrics such as PSNR, SSIM, and LPIPS.

However, as discussed in NoPoSplat [48], reconstructing 3D scenes from sparse unposed views is inherently scale-ambiguous. Although our method successfully generates 3D Gaussians, they may not fully align with the ground-truth scene scale in the validation dataset. To ensure fair comparison with other baselines, we follow pose-free methods by optimizing the target view camera pose while freezing all other parameters. Specifically, we first reconstruct 3D Gaussians, then freeze them and optimize only the target camera pose such that the rendered image closely matches the ground-truth target view. Finally, we compute metrics using the optimized target view camera poses. Note that this optimization scheme is only needed for evaluation purposes and is not required in real-world scenarios.

For the multi-view settings, we additionally perform short test-time optimization following 3DGS [20], denoted as C3G w/ TTO, NoPo+VGGT w/ TTO and AnySplat w/ TTO. We use the predicted Gaussians as initialization and conduct per-scene optimization for a limited number of steps. Specifically, we set the optimization steps to 1,000 steps with densification intervals of 100 steps. We set the learning rate as follows: means at 1.6e-4, scales at 3e-4, rotations at 1e-3, harmonics at 2.5e-3, and densities at 5e-3, using Adam optimizer [21]. Our loss function is formulated as $\mathcal{L} = 0.8\mathcal{L}_{\text{MSE}} + 0.2\mathcal{L}_{\text{SSIM}}$, where $\mathcal{L}_{\text{MSE}}$ denotes the MSE loss and $\mathcal{L}_{\text{SSIM}}$ denotes the SSIM loss between the rendered image and ground-truth images. Other settings follow the original 3DGS. For AnySplat, we follow their default test-time optimization and do not conduct densification to avoid the `Out-Of-Memory` due to the number of Gaussians.

C.2. 3D scene understanding

Datasets. To evaluate 3D scene understanding capabilities, we follow previous approaches [4, 11] and use the ScanNet [9] and Replica [41] datasets.

For ScanNet, we use 40 scenes following LSM [11], selected based on valid pose and depth data where Feature-3DGS [51] performs well. We evaluate 8 categories: wall, floor, ceiling, chair, table, sofa, bed, and others. We select

CVPR #583

30 images with a stride of 10, and target views are chosen as the 1st and 4th images within every 8-image interval.

For Replica, we select 80 images with a stride of 3, and the target view is the 2nd image within every 8-image interval. We use camera poses obtained from COLMAP [38]. We evaluate 3 scenes where LSeg [25] performs well: office3, office4, and room1. We select 5-6 categories per scene. Specifically, for office3, we use wall, ceiling, floor, chair, and table; for office4, we use wall, ceiling, floor, chair, tv-screen, and table; for room1, we use wall, ceiling, floor, bed, and blinds.

Baselines. To validate our effectiveness, we compare our model with per-scene optimized feature novel-view synthesis tasks and feed-forward feature novel-view synthesis.

For per-scene optimization methods (Feature-3DGS [51] and CF^3 [23]), we optimize the per-scene 3D Gaussians using all the posed inputs except for target view images. We then render the 3D Gaussians to target view images to calculate the metrics. For the ScanNet dataset, there are no readily available sparse initial point clouds, so we train with randomly initialized point clouds following the previous works [26, 34]. For feature-3DGS [51], we optimize for 5,000 steps to avoid overfitting. For CF^3 [23], we first optimize 3DGS with 30,000 steps and additionally optimize the CF^3 with 3,000 steps.

For feature novel-view synthesis (LSM [11] and Ours), we first select two input views to generate 3D Gaussians, which are then projected to target views disjoint from the input views. Following the same protocol as novel view synthesis evaluation, these methods require target pose optimization to resolve scale ambiguity.

For LSeg [25] and MaskCLIP [50], we directly extract features from the target viewpoint images, since these methods cannot render features at novel viewpoints like 3D-based approaches.

Evaluation protocol. Given RGB images as input, we first extract language-aware features using models such as LSeg [25] or MaskCLIP [50]. 3D Gaussians with lifted features are generated following each method's procedure, then projected to the target and source views. The target view denotes an unseen viewpoint not present in the input, while the source view denotes a seen viewpoint that every method has observed at least once. We render both RGB images and features at each viewpoint. The rendered features are then processed into segmentation maps by selecting the most relevant CLIP [35] text embeddings corresponding to each label. We report open-vocab segmentation metrics (mIOU and Accuracy), and novel view synthesis metrics (PSNR, SSIM, and LPIPS).

C.3. Multi-view feature encoding

Obtaining multi-view invariant features, also termed as 3D aware features, has been a long-standing goal in computer vision and graphics. Probe3D [10] defines a set of tasks, including two-view correspondences [7, 8, 12–15] and single-view depth estimation [33], to probe the 3D awareness of the features. FiT3D proposes a two-stage fine-tuning task to build view-invariant features, where they first train more than 1000 per-scene feature-3DGS with the desired features. With the pre-trained 3DGS over multiple scenes, they finetune the original vision encoder to follow the rendered features from the 3D Gaussians. They show that after training, they achieve higher correspondence scores in the task defined in Probe3D. As our Comp3D-$\mathcal{F}$ can aggregate the input features, we analyze the effectiveness of Comp3D-$\mathcal{F}$ as a view-invariant feature decoder.

Datasets. To validate the effectiveness of our Comp3D-$\mathcal{F}$ as a multi-view invariant feature decoder, we use the ScanNet [9] dataset following Probe3D [10]. ScanNet is a large-scale dataset of indoor scenes with RGB images, depth maps, and camera poses. We evaluate on 1,500 image pairs from the test split following SuperGlue [37].

Baselines. We compare DINOv2 [31], DINOv3 [39], and VGGT tracking features [42], which are known to perform well on zero-shot correspondence estimation.

Evaluation protocols. Given two images, we first extract a feature map for each image. For ours, we first predict 3D Gaussians with features and project the 3D Gaussians to each input-view pose. Then, we estimate correspondences between the two images using nearest neighbors. Following Probe3D [10], we filter the correspondence using Lowe's ratio test [30] to find the strong unique matches to reduce the noisy correspondence. We rank the correspondences using the ratio test and keep the top 1,000 correspondences.

We evaluate correspondence quality using PCK @ 10px, which measures the accuracy of correspondence within 10 pixels between estimated and ground-truth matches. We divide the image pairs into 4 groups based on viewing angle differences: 0∼15° (θ_0^{15}), 15∼30° (θ_{15}^{30}), 30∼60° (θ_{30}^{60}), and 60∼180° (θ_{60}^{180}). The results can be seen in Tab. 4.

C.4. Multi-view feature upsampling

Feature upsampling is the task of upsampling the extracted features from pre-trained vision encoders to a higher resolution for more fine-grained downstream tasks. Although our Comp3D-$\mathcal{F}$ also takes the low-resolution extracted features from $\mathcal{E}'(\cdot)$ as input, when combined with the 3D Gaussians and rendered to a specific viewpoint, since Gaussians can be rendered at any camera pose, our model can generate features at arbitrary resolutions by setting the desired resolution and intrinsics in the CUDA rasterizer. In this section, we analyze whether the rendered features from Comp3D-$\mathcal{F}$ can be used as an effective multi-view feature upsampler.

Datasets. We use the ScanNet [9] dataset, following the same setup as multi-view feature encoding, to evaluate the probing capability of the upsampled features.

CVPR
#583

Baselines. We compare DINOv2 [31] and DINOv3 [39] features as baselines against their upsampled versions, which match the input image resolution using AnyUp [45], a generalizable feature upsampling module.

Evaluation protocols. Following the same protocol as multi-view feature encoding, we first extract high-resolution feature maps, then estimate correspondences between two images using nearest neighbors. We evaluate correspondence quality using PCK@10px with image pairs divided by viewing angle differences. We also present qualitative results using PCA visualization to demonstrate how upsampled features capture finer details. The results can be seen in Tab. 4.

Table 11. **Comparison of novel view synthesis with compact Gaussians generation strategies.**

	Methods	Average PSNR↑	SSIM↑	LPIPS↓	#G↓
2-view image input	Sampling	21.340	0.665	0.272	**2K**
	Voxelize	19.957	0.609	0.403	4K
	Ours	**22.387**	**0.713**	**0.259**	**2K**
24-view image input	Sampling	22.446	0.714	0.219	25K
	Sampling w/ TTO	25.487	0.832	0.158	28K
	Voxelize	19.052	0.558	0.446	81K
	Voxelize w/ TTO	27.751	0.885	0.165	90K
	NoPo+VGGT	21.244	0.664	0.200	1,204K
	NoPo+VGGT w/ TTO	28.463	0.902	0.135	1,204K
	Ours	23.797	0.747	0.198	**2K**
	Ours w/ TTO	**29.987**	**0.916**	**0.136**	27K

Table 12. **Analysis of tradeoff between rendering performance and number of Gaussians.**

# of Gaussians	PSNR↑	SSIM↑	LPIPS↓	FPS↑
2048×1	20.625	0.623	0.321	**3083.50**
2048×2	20.844	0.639	0.303	3015.03
2048×4	21.080	0.644	0.297	2828.14
2048×8	**21.218**	**0.653**	**0.284**	2476.01

D. Additional ablation and discussion

D.1. Comparison of compact 3D reconstruction strategies

To validate the effectiveness of our compact reconstruction strategy, we conduct ablation studies comparing different approaches to obtain compact representations. For the sampling baseline, we first estimate per-pixel Gaussian centers and their corresponding Gaussians, then downsample by $8\times$ to predict only 2,048 Gaussians, matching the number of Gaussians with our method. For the voxelization baseline, we follow AnySplat's [19] strategy: we first estimate per-pixel Gaussians, then voxelize them with additional layers. Since AnySplat's original voxel size cannot sufficiently reduce the number of Gaussians, we increase the voxel size to 0.2, which results in approximately 4K Gaussians to best match the 2,048 Gaussians produced by our method.

As shown in Tab. 11, our strategy demonstrates superior performance compared to sampling or voxelization methods while using fewer Gaussians than voxelization. Both sampling and voxelization rely on per-pixel estimation, which restricts them to pixel locations and prevents accurate estimation of necessary regions. Additionally, voxelization requires heuristic determination of voxel size and can reduce details in high-frequency regions. In contrast, our approach does not depend on pixel locations as learnable tokens can flexibly estimate Gaussians at appropriate locations, enabling more effective compact Gaussian field reconstruction.

Table 13. **Additional ablation studies for Comp3D-$\mathcal{F}$.**

Detach attn.	Detach $\mathcal{L}_{\text{feat}}$ to G_i	Lseg [25] mIOU↑	Acc.↑	PSNR↑	SSIM↑	LPIPS↓
✗	✓	0.490	0.761	23.078	0.750	0.293
✓	✗	0.490	0.757	23.243	0.754	0.286
✓	✓	**0.513**	**0.783**	**23.886**	**0.770**	**0.285**

D.2. Additional analysis of number of Gaussians

We first analyze the tradeoff between rendering performance and the number of Gaussians in Tab. 6. Notably, this experiment reveals a slight performance drop when increasing the number of learnable tokens from 2,048 to 4,096. We attribute this to our framework's design of using a fixed number of learnable queries, which encourages each query to learn cross-view correspondences. Increasing the number of tokens appears to weaken this inductive bias toward coherent multi-view aggregation, making attention scores less localized and yielding no additional benefit. In contrast, our experiments in Tab. 12 demonstrate that increasing the number of decoded Gaussians *per* token consistently improves performance, enabling the model to capture fine-grained details from scenes coarsely represented by 2K tokens. We believe this analysis highlights a promising direction for applications that prioritize feed-forward NVS performance. Note that the results in Tab. 12 are reported after 75K training steps (out of 450K) to assess feasibility.

D.3. Additional ablation for feature decoder

We additionally ablate the components of Comp3D-$\mathcal{F}$. In Tab. 13, we conduct experiments on (1) detaching the copied attention weights from Comp3D-$\mathcal{G}$, and (2) propagating feature loss to Gaussian geometry attributes (means, covariances, spherical harmonics, and opacities). When feature loss propagates to the copied attention weights, Comp3D-$\mathcal{G}$ receives ambiguous gradients because the features are not perfectly multi-view consistent. Consequently, the attention mechanism cannot correctly identify correspondences between learnable query tokens and the encoder features $\mathcal{E}(\cdot)$ (e.g., VGGT) for Comp3D-$\mathcal{G}$. To propagate feature loss to Gaussian attributes, we modify the CUDA rasterizer from Feature-3DGS [51], which originally propagates feature loss only to Gaussian feature attributes. We extend the CUDA rasterizer to propagate feature loss to all Gaussian attributes. However, this also degrades geometry estimation results because foundation model features are not perfectly multi-view consistent. We hypothesize that with perfectly multi-view invariant features, feature loss could improve reconstruction quality, especially since photometric loss is also imperfect, as real-world RGB images contain visual artifacts such as appearance variations, lighting changes, and noise.

CVPR #583

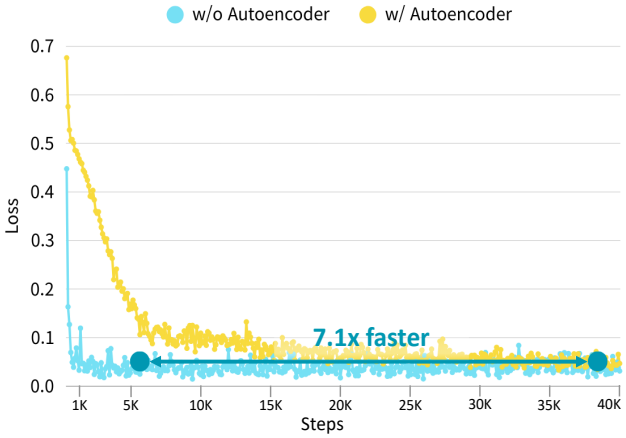


Figure 6. **Convergence speed improvement by eliminating autoencoder in our framework.**

Table 14. **Computational Cost.**

	Methods	Inference time (s) ↓	GPU Mem. (MiB) ↓	FPS ↑
2-view image input	NoPoSplat [48]	0.092	6,063	898.28
	NoPo+VGGT	0.093	7,495	1566.28
	Ours	**0.091**	**4,150**	**2742.68**
24-view image input	AnySplat [15]	1.449	6,722	225.50
	AnySplat w/ TTO [15]	35.907	6,722	225.50
	Ours	**0.466**	**4,372**	**3083.50**
	Ours w/ TTO	4.273	**4,372**	2511.49
Feature lifting	LSM [9]	0.165	**5,333**	625.12
	Ours	**0.093**	5,896	**785.35**

D.4. Discussion of autoencoder in existing methods

To reduce the memory consumption of Gaussians, prior works [11, 23] commonly adopt autoencoders to compress Gaussian feature dimensions and restore them via a decoder. However, autoencoder-based compression inevitably introduces information loss. Moreover, as shown in Fig. 6, removing the autoencoder allows our model to converge **7.1**× faster. With an autoencoder framework, additional training time is required for the encoder–decoder architecture. In contrast, without an autoencoder, we directly leverage the attention mechanism of Comp3D-$\mathcal{G}$ to aggregate and store features. Since the attention maps in Comp3D-$\mathcal{G}$ already identify the necessary features for each learnable token, each feature token naturally attends to the same multi-view regions as its corresponding Gaussian token, effectively reusing learned correspondences for feature aggregation. This emergent property enables highly efficient feature lifting with minimal additional training overhead.

D.5. Discussion of Computational Costs

We include a detailed analysis of computational costs in Tab. 14. In *novel-view synthesis* settings, compared to prior works [19, 48] that employ a DPT head — which incurs significant computational overhead due to the large feature resolution space — our shallow transformer decoder achieves faster inference with lower peak GPU memory usage. For *feature lifting*, our model achieves higher rendering speed and FPS compared to LSM [11]. While LSM must compress features to remain tractable due to its large number of Gaussians, our compactness enables feature lifting and rendering without any compression. Although increasing the feature dimension slightly raises peak GPU memory, the overhead is modest and predictable, enabling our method to deliver higher PSNR and mIoU — validating that compact Gaussians provide an efficient and effective substrate for feature lifting.

D.6. Discussion of FPS

As shown in Tab. 3 and Tab. 4, our method reduces the number of Gaussians by **65**× (to only **2K**) compared to per-pixel approaches [11, 48], yet FPS does not scale linearly with this reduction. We further analyze FPS in Tab 14. In the main table, FPS measurements include camera matrix multiplication and tensor operations for pre-processing prior to Gaussian rasterization. We additionally report FPS measured over the rendering step alone, which demonstrates that our compactness becomes more valuable when isolating the rendering stage, yielding substantially higher FPS.

D.7. Limitations

Although our view-invariant feature decoder, Comp3D-$\mathcal{F}$, allows lifting arbitrary 2D features into 3D, we have not evaluated all recent foundation models. For instance, we believe that leveraging features from the Segment Anything Model (SAM) [22] could enable robust multi-view consistent segmentation if the features are aggregated within our framework before being decoded by a pre-trained SAM decoder; however, we did not analyze these specific features in this work.

In addition, following prior works, our experimental validation primarily focuses on multi-view open-vocabulary segmentation. To achieve holistic 3D scene understanding, future work could explore integrating our feature fields with recent Multimodal Large Language Models (MLLMs) [28, 29] to enable 3D Scene Question Answering. Furthermore, since our framework enables novel view feature rendering without information loss or compression artifacts, it holds significant promise for integration with Vision-Language-Action (VLA) models or robotics applications. However, such scenarios are frequently dynamic, whereas our current framework is limited to static scene reconstruction. Extending our compact representation to dynamic scenes remains an exciting avenue that would unlock potential across various autonomous fields.

CVPR #583

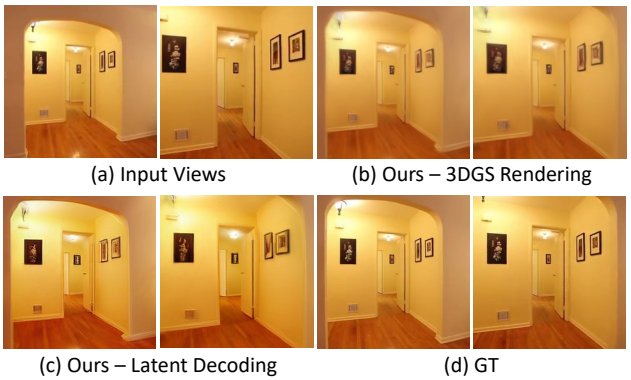


Figure 7. **Novel view synthesis via latent decoding.** We explore the potential of combining our view-invariant feature decoder, Comp3D-$\mathcal{F}$, with generative models. We lift DINOv2-base features (which serve as latents for a Representation Autoencoder (RAE) [49]) extracted from the input views ((a)) to 3D Gaussians and render them at novel viewpoints. (b) **Ours – 3DGS Rendering**: Standard RGB rendering from the estimated Gaussians provides faithful reconstruction but exhibits some blurriness in high-frequency regions. (c) **Ours – Latent Decoding**: By decoding the rendered feature maps through the RAE's diffusion transformer with a 1-step denoising process, we recover significantly sharper textures and edges. Although the generative nature introduces slight variations in fine details compared to the Ground Truth (d), this validates our model's ability to provide consistent 3D-aware latents for diffusion-based pipelines.

E. Additional experiment results

E.1. Novel view synthesis via latent decoding

As our multi-view invariant feature decoder Comp3D-$\mathcal{F}$ can take any desired feature as input, we conduct an additional experiment of leveraging the latents of the representation autoencoder (RAE) [49] as input. Specifically, we leverage the open-sourced RAE, which learns a diffusion transformer [32] and a decoder based on the frozen DINOv2-base [31] as the latent. By lifting the DINOv2-base features and rendering novel viewpoints, we can decode the rendered features using the pre-trained RAE decoder or DiT after adding a small noise to the rendered features. In Fig. 7, we show the results of novel view synthesis obtained from the estimated 3D Gaussians (Fig. 7-(b)) and the novel view synthesis obtained by decoding the rendered features through the DiT (Fig. 7-(c)). Specifically, for this experiment, we add a 1-step noise from the original 50-step denoising schedule of RAE-DiT [49], and denoise with 1-step. As shown in Fig. 7, renderings from 3D Gaussians already provide sufficiently good results but still contain some blurry regions, whereas the latent decoding yields significantly sharper results. However, since the pipeline involves a generative process, even with a single step, we observe slight deformations in fine details. Nevertheless, this experiment demonstrates the potential of combining novel view synthesis with diffusion processes by leveraging latents as input to our Comp3D-$\mathcal{F}$.

Table 15. **Comparison of novel view synthesis on RealEstate10K [52].** Our method maintains competitive results while using far fewer Gaussians.

Pose-free	Methods	Average					
		#G ↓	Memories↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓
✗	PixelSplat [5]	131K	33.6MB	388.03	23.848	0.806	0.185
	MVSplat [6]	131K	33.6MB	392.6	23.977	0.811	0.176
✓	CoPoNeRF [17]	-	-	0.4	18.938	0.619	0.388
	Splatt3R [40]	131K	33.6MB	393.1	15.318	0.490	0.436
	PF3plat [16]	131K	33.6MB	397.1	21.042	0.739	0.233
	SPFSplat [18]	131K	33.6MB	397.3	25.845	0.852	0.152
	NoPoSplat [48]	131K	33.6MB	369.8	25.033	0.838	0.160
	VGGT+NoPo [42, 48]	100K	25.6MB	419.8	23.015	0.762	0.187
	AnySplat [19]	320K	81.9MB	319.2	18.828	0.656	0.358
	C3G (Ours)	2K	0.1MB	451.7	22.387	0.713	0.259

E.2. Novel view synthesis in two-view setting

In this section, we evaluate novel view synthesis performance on the RealEstate10K dataset [52], following NoPoSplat's [48] protocol where two input images are used to estimate 3D Gaussians and render a target view. We compare with both *pose-dependent* models [5, 6] and *pose-free* models [16–18, 40, 48]. Since we use VGGT [42] as our visual encoder for Comp3D-$\mathcal{G}$, we additionally train VGGT+NoPo, which replaces NoPoSplat's MASt3R [24] backbone with VGGT [42] while maintaining NoPoSplat's pipeline to estimate per-pixel Gaussians. Note that Comp3D-$\mathcal{G}$ directly estimates Gaussians from unposed images, falling into the *pose-free* category. For NoPoSplat, VGGT+NoPo, and ours, we follow NoPoSplat's test-time camera pose optimization, which is only necessary for evaluation. As shown in Tab. 15, despite estimating **65**× fewer Gaussians than per-pixel methods [16–18, 40, 48], our approach achieves comparable rendering quality with much faster speeds, validating that our compact Gaussians is sufficient for 3D scene reconstruction.

F. Additional qualitative results

F.1. Additional results of attention map visualization

We additionally visualize the attention maps between learnable tokens $\mathbf{Q}$ and features $\mathbf{F}$ in $\mathcal{T}_{\mathcal{G}}$, extending Fig. 3-(b). As illustrated in Fig. 8, Comp3D-$\mathcal{G}$ exhibits sharp, focused attention patterns on spatially coherent regions across multiple views for all Gaussian tokens and input images. These results demonstrate an emergent behavior: to accurately reconstruct novel views with a limited number of N Gaussians, the model learns to position 3D Gaussians at geometrically coherent regions.

CVPR
#583

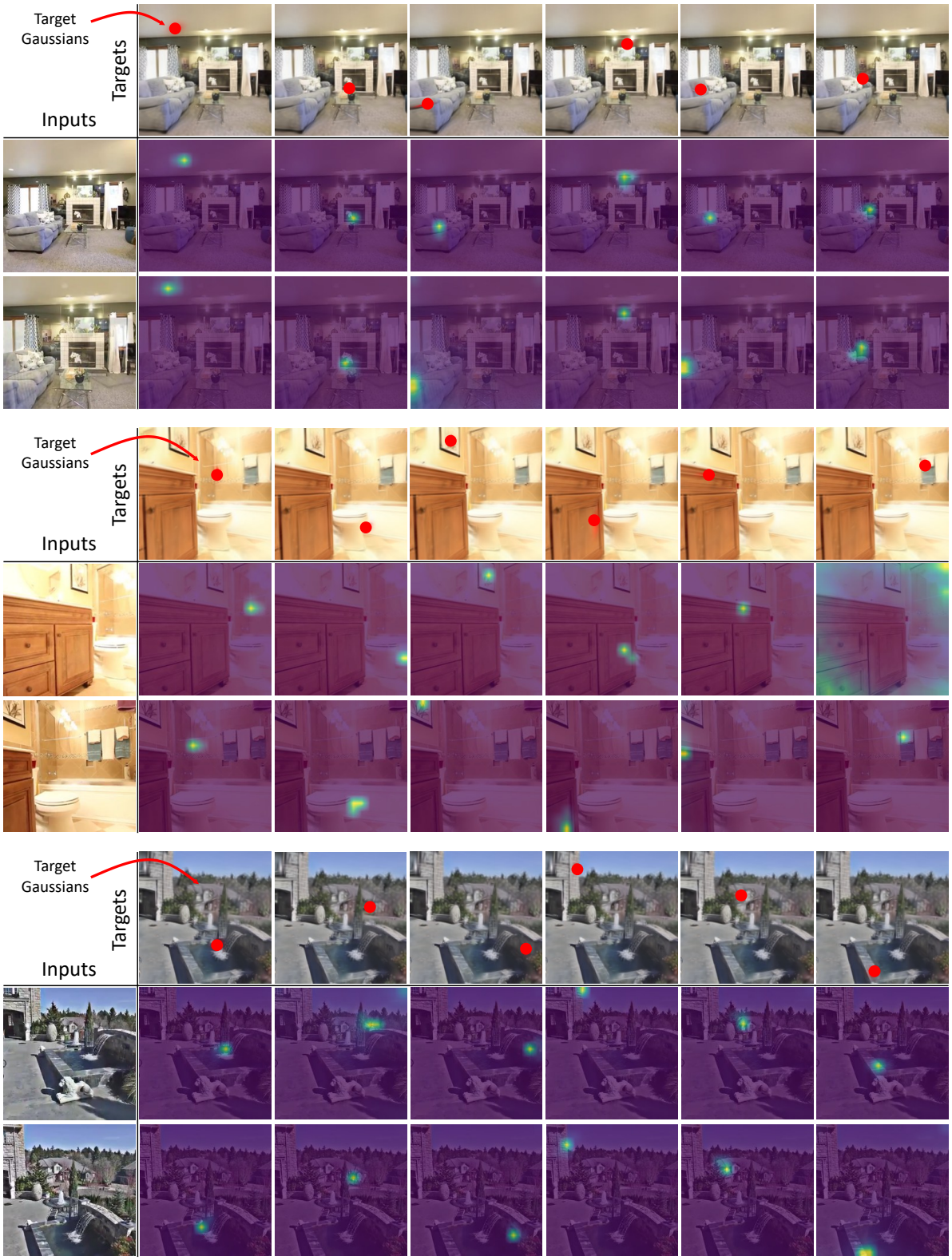


Figure 8. **Additional visualization of learned attention patterns between a target Gaussian and input images.** Without explicit supervision, each query token (red dots) learns to attend to spatially coherent regions across multiple views, naturally discovering corresponding regions.

CVPR
#583

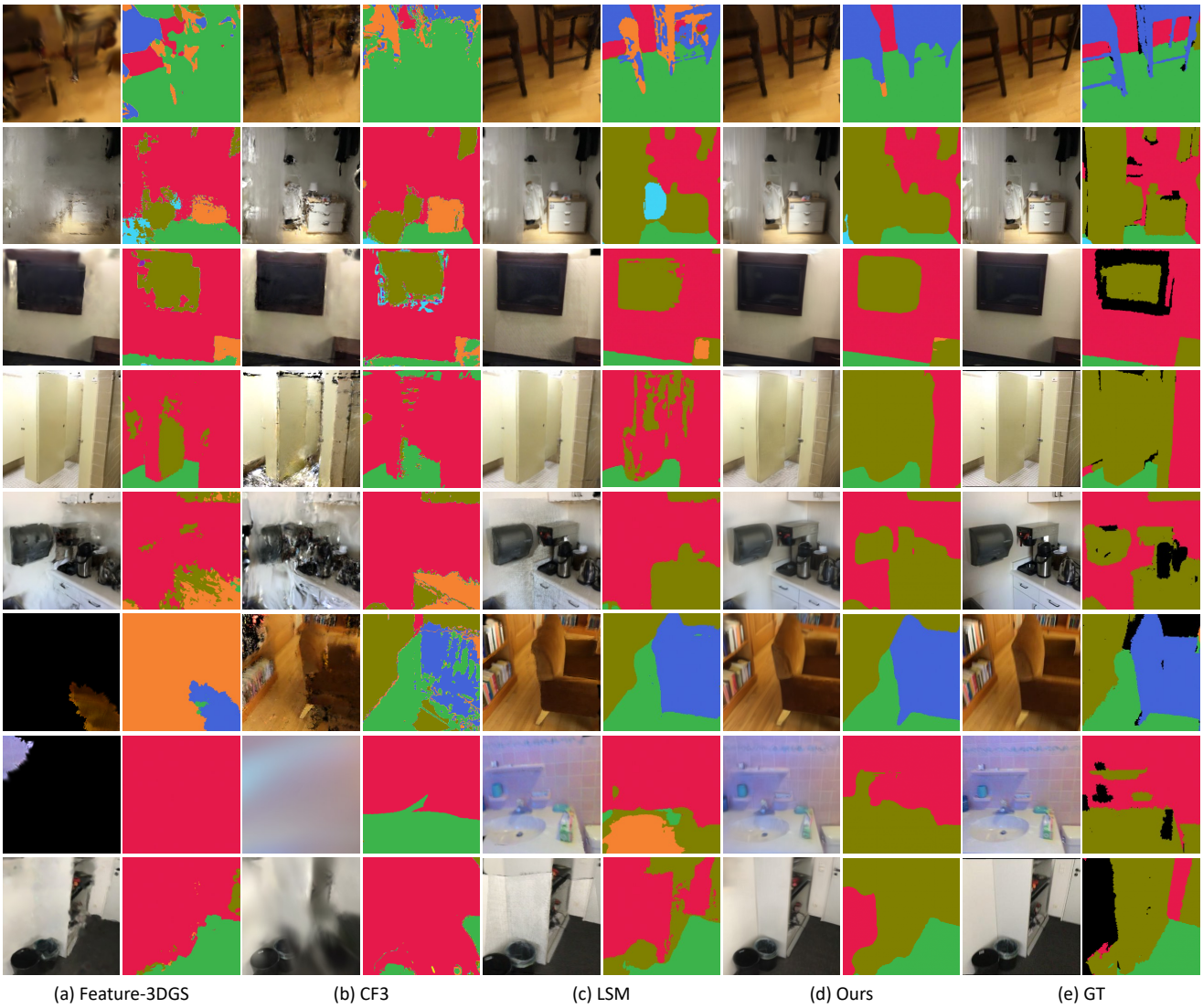


Figure 9. **Additional qualitative results of 3D scene understanding on ScanNet [9].** We conduct qualitative comparison for 3D scene understanding via novel view synthesis and open-vocabulary segmentation. When compared to both per-scene optimization ((a), (b)) and feed-forward ((c), (d)) methods, ours show the most high-fidelity renderings and accurate segmentation maps compared to the ground-truth.

F.2. Additional qualitative results of 3D scene understanding

We additionally present qualitative results for 3D scene understanding on the ScanNet dataset. In Fig. 9, we visualize novel view synthesis results and open-vocabulary segmentation results from feature-lifted 3D Gaussians. Our method generates more geometrically accurate 3D Gaussians and more effectively aggregates multi-view features than competing methods.

F.3. Qualitative results of multi-view feature encoding

In Fig. 10, we additionally visualize the comparison between pre-lifting features and features aggregated by Comp3D-$\mathcal{F}$ using PCA. As shown, our model effectively aggregates multi-view features before lifting, producing representations that are both view-invariant and more semantically discriminative.

F.4. Qualitative results of multi-view feature upsampling

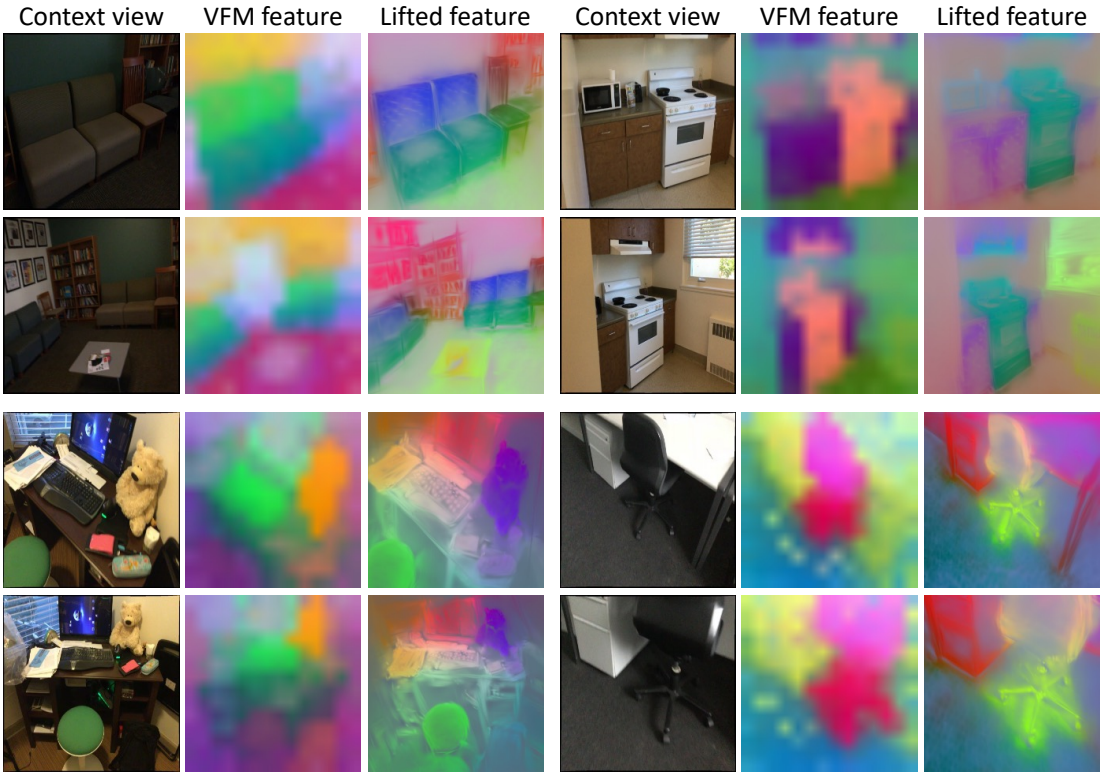
In Fig. 11, we present qualitative results of multi-view feature upsampling results.

CVPR
#583

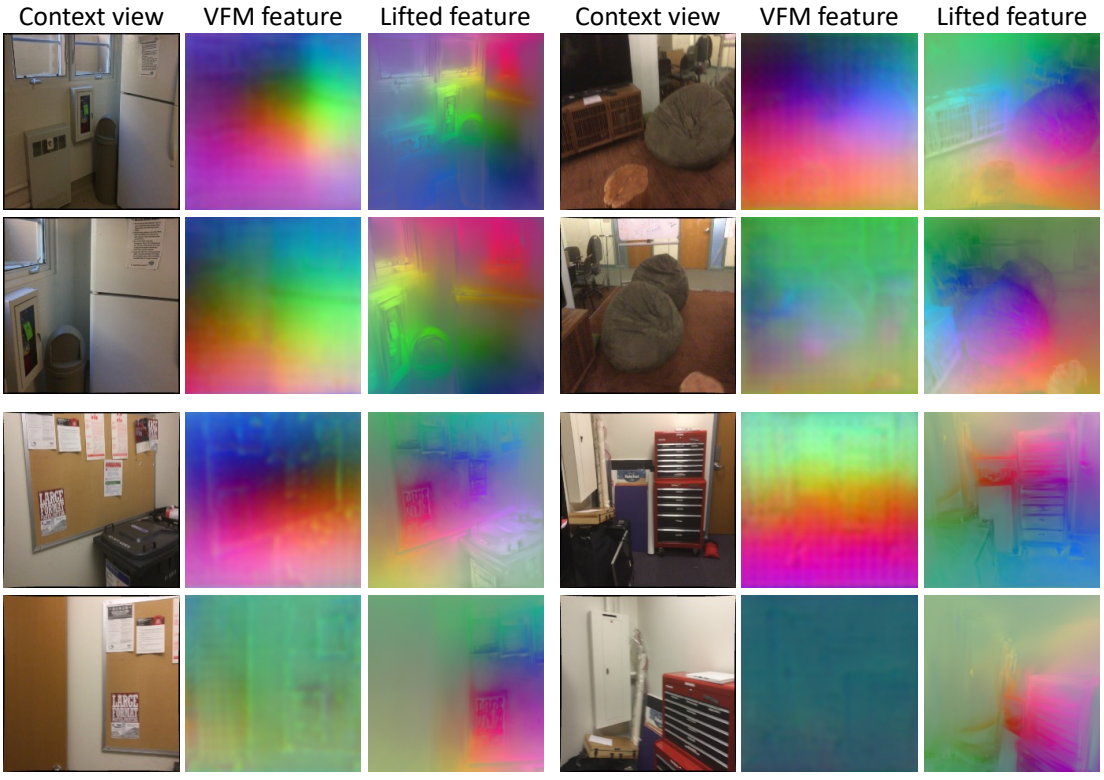
F.5. Qualitative results of novel view synthesis with multi-view inputs

In Fig. 12, we provide qualitative comparisons of novel view synthesis using 24 input images. We compare our method against AnySplat [19], a state-of-the-art feed-forward approach that supports arbitrary numbers of input views. Despite using significantly fewer Gaussians (approx. 2K) compared to AnySplat (approx. 2.6M), our method produces more geometrically consistent renderings with higher quality renderings ((b) vs. (d)). Furthermore, when applying short test-time optimization (denoted as Ours w/ TTO and AnySplat w/ TTO), our method significantly outperforms AnySplat in recovering high-frequency details, demonstrating that our compact representation is sufficient for reconstruction and novel view synthesis while also serving as a robust initialization for per-scene optimization.

CVPR
#583



(a) DINOv3



(b) VGGT-Tracking

Figure 10. **Additional PCA visualization of multi-view features on ScanNet [9].** We visualize the PCA results of encoded multi-view features. Our method improves multi-view consistency compared to the original visual features.

CVPR
#583

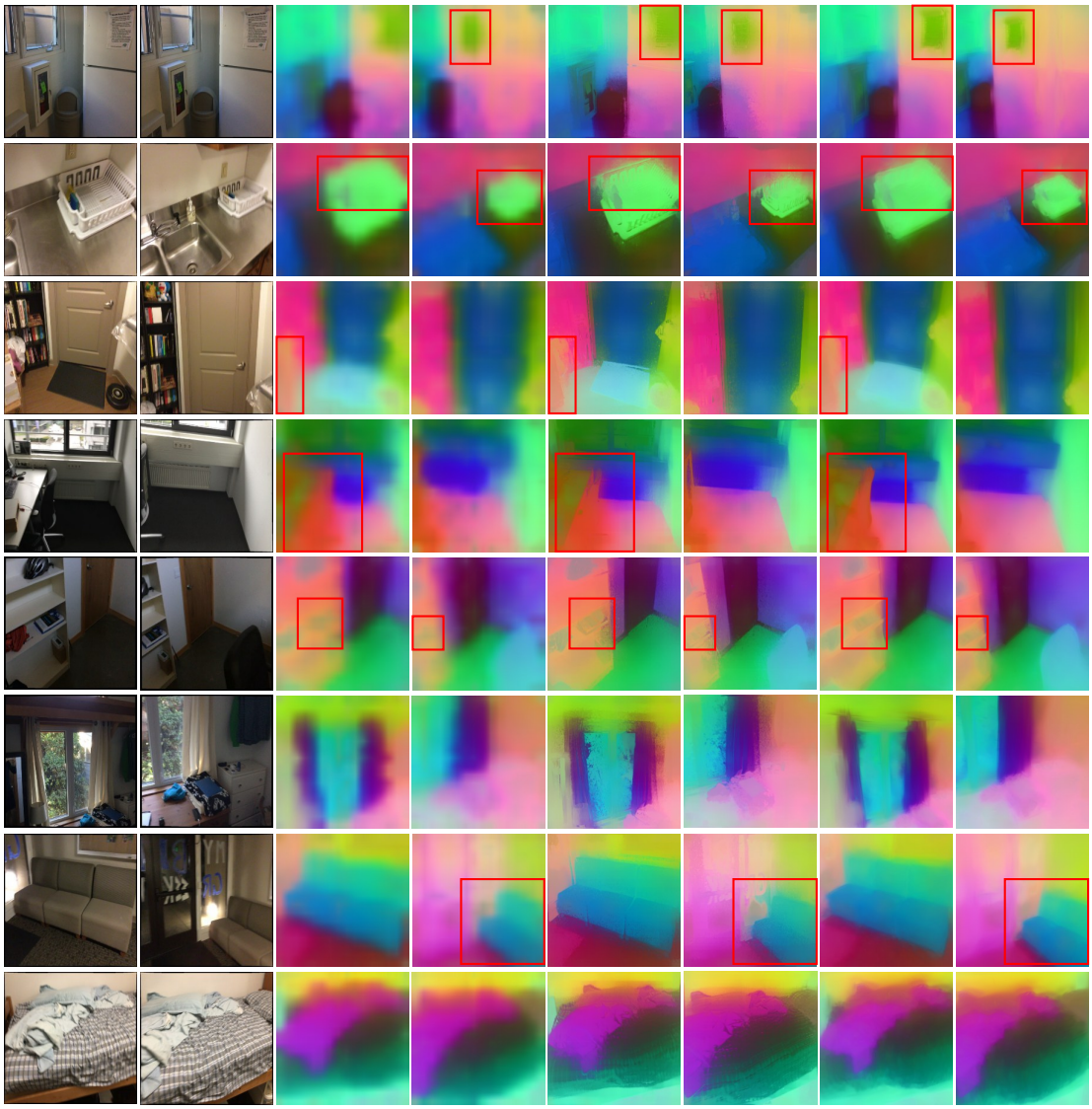


Figure 11. **Additional PCA visualization of upsampled feature on ScanNet [9].** We visualize the PCA results of the upsampled feature. Our model upsamples the features while maintaining the multi-view consistencies compared to other baselines.

CVPR
#583

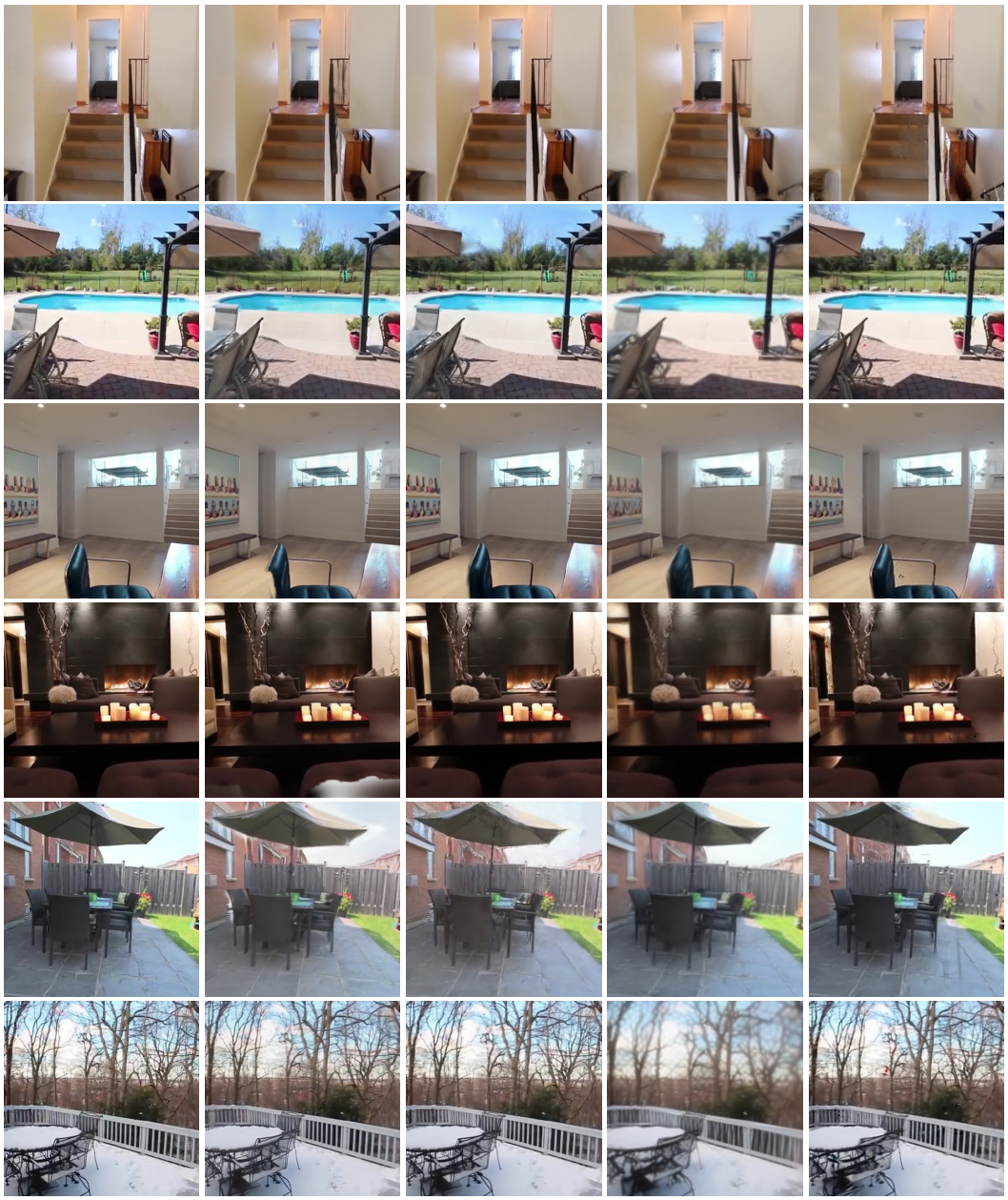


Figure 12. **Additional qualitative results of novel view synthesis on RealEstate10K [52].** We conduct a qualitative comparison for novel view synthesis with available multi-view images. Our method produces the highest quality rendering results, with or without test-time Gaussian optimization. TTO denotes that test-time optimization is applied to the Gaussians.

References

[1] Eduardo Arnold, Jamie Wynn, Sara Vicente, Guillermo Garcia-Hernando, Aron Monszpart, Victor Prisacariu, Daniyar Turmukhambetov, and Eric Brachmann. Map-free visual relocalization: Metric pose relative to a single image. In *European Conference on Computer Vision*, pages 690–708. Springer, 2022. 2

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021. 2

[4] Neil Burgess. Spatial memory: how egocentric and allocentric combine. *Trends in cognitive sciences*, 10(12):551–557, 2006. 3

[5] 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*, pages 19457–19467, 2024. 3, 8

[6] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2024. 3, 8

[7] Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. *Advances in Neural Information Processing Systems*, 34:9011–9023, 2021. 4

[8] Seokju Cho, Sunghwan Hong, and Seungryong Kim. Cats++: Boosting cost aggregation with convolutions and transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7174–7194, 2022. 4

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 3, 4, 10, 12, 13

[10] Mohamed El Banani, Amit Raj, Kevis-Kokitsi Maninis, Abhishek Kar, Yuanzhen Li, Michael Rubinstein, Deqing Sun, Leonidas Guibas, Justin Johnson, and Varun Jampani. Probing the 3d awareness of visual foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21795–21806, 2024. 4

[11] Zhiwen Fan, Jian Zhang, Wenyan Cong, Peihao Wang, Renjie Li, Kairun Wen, Shijie Zhou, Achuta Kadambi, Zhangyang Wang, Danfei Xu, et al. Large spatial model: End-to-end unposed images to semantic 3d. *Advances in neural information processing systems*, 37:40212–40229, 2024. 3, 4, 7

[12] Sunghwan Hong and Seungryong Kim. Deep matching prior: Test-time optimization for dense correspondence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9907–9917, 2021. 4

[13] Sunghwan Hong, Seokju Cho, Jisu Nam, Stephen Lin, and Seungryong Kim. Cost aggregation with 4d convolutional swin transformer for few-shot segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2022.

[14] Sunghwan Hong, Jisu Nam, Seokju Cho, Susung Hong, Sangryul Jeon, Dongbo Min, and Seungryong Kim. Neural matching fields: Implicit representation of matching fields for visual correspondence. *Advances in Neural Information Processing Systems*, 35:13512–13526, 2022.

[15] Sunghwan Hong, Seokju Cho, Seungryong Kim, and Stephen Lin. Unifying feature and cost aggregation with transformers for semantic and visual correspondence. *arXiv preprint arXiv:2403.11120*, 2024. 4

[16] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024. 3, 8

[17] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jiaolong Yang, Seungryong Kim, and Chong Luo. Unifying correspondence pose and nerf for generalized pose-free novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20196–20206, 2024. 3, 8

[18] Ranran Huang and Krystian Mikolajczyk. No pose at all: Self-supervised pose-free 3d gaussian splatting from sparse views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27947–27957, 2025. 3, 8

[19] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *arXiv preprint arXiv:2505.23716*, 2025. 1, 2, 3, 6, 7, 8, 11

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 3

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 3

[22] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 7

[23] Hyunjoon Lee, Joonkyu Min, and Jaesik Park. Cf3: Compact and fast 3d feature fields. *arXiv preprint arXiv:2508.05254*, 2025. 4, 7

[24] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024. 3, 8

[25] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 4, 6

[26] Wanhua Li, Yujie Zhao, Minghan Qin, Yang Liu, Yuanhao Cai, Chuang Gan, and Hanspeter Pfister. Langsplatv2: High-dimensional 3d language gaussian splatting with 450+ fps. *arXiv preprint arXiv:2507.07136*, 2025. 4

[27] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 2

[28] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023. 7

[29] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024. 7

[30] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 4

[31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4, 5, 8

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 8

[33] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024. 4

[34] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 4

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 4

[36] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021. 2

[37] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 4

[38] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3, 4

[39] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 2, 4, 5

[40] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024. 3, 8

[41] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 3

[42] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 1, 2, 3, 4, 8

[43] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5261–5271, 2025. 1

[44] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 2

[45] Thomas Wimmer, Prune Truong, Marie-Julie Rakotosaona, Michael Oechsle, Federico Tombari, Bernt Schiele, and Jan Eric Lenssen. Anyup: Universal feature upsampling. *arXiv preprint arXiv:2510.12764*, 2025. 5

[46] Hongchi Xia, Yang Fu, Sifei Liu, and Xiaolong Wang. Rgbd objects in the wild: Scaling real-world 3d object learning from rgb-d videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22378–22389, 2024. 2

[47] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1790–1799, 2020. 2

[48] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024. 1, 2, 3, 7, 8

[49] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025. 8

[50] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *European conference on computer vision*, pages 696–712. Springer, 2022. 4

CVPR
#583

[51] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta 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*, pages 21676–21685, 2024. 3, 4, 6

[52] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1, 3, 8, 14