FreeSplatter: Pose-free Gaussian Splatting for Sparse-view 3D Reconstruction

Supplementary Material

In this supplementary material, we present additional implementation details (Section 1) and experimental results (Section 2).

1. Implementation Details

Architecture and Training Details. FreeSplatter-O and FreeSplatter-S share an identical model architecture consisting of a 24-layer transformer with a width of 1024, totaling approximately 306 million parameters. Bias terms are omitted throughout the model, except in the output linear layer for Gaussian prediction. The models are initially trained on 256×256 input images for 500K steps, followed by fine-tuning on 512×512 input images for 100K steps, using 16 NVIDIA H800 GPUs. Training employs the AdamW [6] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a weight decay of 0.01, and a learning rate of 4×10^{-5}. The maximum pre-training step is set to $T_{\max} = 10^5$, meaning the Gaussian position loss in Equation 5 is applied during the first 100K training steps. To improve training efficiency, we leverage the xFormers [5] library for memory-efficient attention, gradient checkpointing [1], bfloat16 mixed-precision training [7], and deferred backpropagation [15] for GS rendering. For each batch, we sample 4 input views for FreeSplatter-O and 2 for FreeSplatter-S, along with 4 novel views for supervision.

Camera Intrinsic Estimation. Following DUSt3R [11], we assume centered principle point and square pixels in this work. Thanks to the point-cloud-based Gaussian representation, we can extract the Gaussian locations as "point maps" $\left\{\boldsymbol{X}^n \in \mathbb{R}^{H\times W\times 3} \mid n = 1, \ldots, N\right\}$ from the predicted Gaussian maps. Since all points are predicted in the first view's camera frame, we can estimate the focal length of the first view by minimizing the pixel-projection errors:

$$f^1_* = \arg\min_{f^1} \sum_{i=0}^{W-1}\sum_{j=0}^{H-1} \left\| (i', j') - f^1 \frac{(X^1_{i,j,0}, X^1_{i,j,1})}{X^1_{i,j,2}} \right\|, \tag{1}$$

where $(i', j') = (i - \frac{W}{2}, j - \frac{H}{2})$ denotes pixel coordinates relative to the principle point $(\frac{H}{2}, \frac{W}{2})$. This optimization problem can be solved by Weiszfeld algorithm [8] efficiently.

For multi-view scenarios, we cyclically translate the input views so that each view becomes the "first view" in turn, and estimate the focal length for each individually. Finally, we compute the average focal length and use it for all views: $f = \frac{1}{N}\sum_{n=1}^{N} f^n$.

Masks for PnP-RANSAC solver. We adopt different masks $\boldsymbol{M}^n$ in Equation 4 for object-centric and scene-level reconstruction when estimating the input camera poses:

- For object-centric reconstruction, the model is trained on white-background images rendered from 3D assets, and the predicted Gaussians at the background area are not restricted to be pixel-aligned (See Section 3.3 for details), which may influence the camera pose solving. Therefore, we assume white-background input images at inference time, and adopt the foreground mask segmented by an off-the-shelf background-removal tool[1] as $\boldsymbol{M}^n$. The segmentation masks are not always perfect but are precise enough for the PnP-RANSAC solver to estimate the poses robustly.
- For scene-level reconstruction, we do not distinguish the foreground and background areas during training, and all the predicted Gaussians are restricted to be pixel-aligned by applying the pixel-alignment loss universally. Therefore, all visible Gaussians should contribute to pose solving and we use the Gaussian opacity map $\boldsymbol{O}^n \in \mathbb{R}^{H\times W}$ to compute $\boldsymbol{M}^n$ with a minimal visibility threshold τ: $\boldsymbol{M}^n = (\boldsymbol{O}^n > \tau)$.

Camera Normalization. The definition of reconstruction frame plays an important role in model training. We take the camera frame of the first view as the reference frame for reconstruction, and normalize all cameras in a training batch into this frame. Besides, a scaling operation is required to deal with various scene scales. Denoting the original input cameras as $\{\boldsymbol{P}^n_{\text{in}} = [\boldsymbol{R}^n_{\text{in}} \mid \boldsymbol{t}^n_{\text{in}}] \mid n = 1, \ldots, N\}$, we adopt different camera normalization strategies for object-centric and scene-level reconstruction.

For object-centric reconstruction, the training cameras are sampled orbiting the object center. We first scale all input cameras to make the distance from the first camera to the object center equal to 2, and then transform all cameras into the reference frame so that the first view's camera pose is an identity matrix. With this strategy, we fix the object center at $(0, 0, 2)$ in the reference frame. For scene-level reconstruction, the camera distributions are more complex. Thus we fist transform all cameras into the reference frame, and then scale them using a factor $s = 1/d$, d is the mean distance of all valid points to the origin:

$$d = \frac{1}{\sum_{n=1}^{N} |\boldsymbol{M}^n|} \sum_{n=1}^{N}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} M^n_{i,j} \cdot \left\|\boldsymbol{X}^n_{i,j}\right\|, \tag{2}$$

where $\boldsymbol{X}^n \in \mathbb{R}^{H\times W\times 3}$ denotes the ground truth point map unprojected from depth map, $\boldsymbol{M}^n \in \mathbb{R}^{H\times W}$ is the valid depth mask. To be noted, we also scale the ground truth depths when scaling the cameras.

[1] https://github.com/danielgatis/rembg

Scene Scale Ambiguity. Given a set of scene images, we can not infer the actual scale of the scene, since we can obtain identical rendered images when scaling the scene and cameras simultaneously, *i.e.*, the scene scale ambiguity problem. For object-centric reconstruction, we have fixed the object center at $(0, 0, 2)$ in the reference frame, so there is no scale ambiguity problem. But for scene images, it is hard to define a canonical center and it is reasonable for our model to reconstruct the scene in arbitrary scale. Besides, the training data is a mixture of multiple datasets that vary greatly in scale, which may make the model confused on scale prediction. Then a question arises, how can we align the scale of the model's prediction with the scale of the training cameras so that we can render target views for supervision?

As mentioned above, we scale the camera locations in a training batch using $s = 1/d$ (d is defined in Equation 2), which can bound the scene into a range reasonable for the model to predict. However, the scale factor is related to the choice of valid points, leading to different scene scales with a minimal difference in selected points. The model would get confused if we train it to reconstruct exactly in this scale since it has no idea which points we utilized to compute this scale. Inspired by DUSt3R [11], we choose to rescale the reconstructed Gaussians before rendering them to deal with scene ambiguity. We first compute the scale factor $\hat{s} = 1/\hat{d}$ using Equation 2 with predicted Gaussian positions $\hat{\boldsymbol{X}}^n$ and the same mask $\boldsymbol{M}^n$. Then we adjust the *position* and *scale* of each Gaussian as $\hat{\boldsymbol{\mu}}_k = \hat{s} \cdot \boldsymbol{\mu}_k, \hat{\boldsymbol{s}_k} = \hat{s} \cdot \boldsymbol{s}_k$. With this operation, we do not care about the absolute scale of the reconstructed scene, but we expect it to align with the ground truth scene after scaling them using the factors $\hat{s}$ and s respectively. In the pre-training stage, we compute $\mathcal{L}_{\text{pos}}$ using the scaled predicted positions and ground truth positions too.

2. Additional Experiments

2.1. Comparison with PF-LRM

We include additional qualitative comparison with PF-LRM in Figure 2. PF-LRM [10] is the most relevant work to FreeSplatter-O. Although both of them deal with pose-free sparse-view reconstruction and adopt a feed-forward transformer architecture, there are fundamental distinctions between them:

- Although PF-LRM is pose-free, it still requires ground truth camera intrinsics as input. FreeSplatter operates without camera poses or intrinsics, enhancing its practical applicability since it is very challenging to obtain the camera intrinsics in many scenarios.
- In this work, we hope to point out that, *Gaussian Splatting is inherently more suitable for joint 3D reconstruction and camera pose estimation*. The point-based nature of Gaussian Splatting enables direct application of PnP algorithms. By predicting multi-view Gaussian maps as "augmented" point maps, our model facilitates both high-quality 3D modeling and accurate pose estimation. In contrast, PF-LRM's triplane NeRF architecture requires: (i) separate branches for reconstruction (triplane tokens $\rightarrow$ triplane NeRF) and pose estimation (image tokens $\rightarrow$ point clouds), and (ii) additional training overhead due to triplane token processing. FreeSplatter-O directly maps image tokens to Gaussians which handle both tasks, thus is more elegant.
- Triplane NeRF faces challenges in modeling complex scenes due to its limited resolution, restricted background modeling capability, and memory-intensive volume rendering. This means that training a scene-level PF-LRM to achieve the similar function and performance of FreeSplatter-S will be very challenging. In comparison, FreeSplatter-S demonstrates superior scene reconstruction while maintaining the same architectural framework as FreeSplatter-O. We believe this scalability represents a significant advancement toward open-world high-quality reconstruction.

We also hope to discuss the quantitative results on PF-LRM's GSO evaluation datasets. As Table 1 and Table 2 of the main paper shows, PF-LRM achieves better metrics than FreeSplatter-O on its GSO evaluation dataset. This is in fact due to the failure cases caused by the lighting of images. PF-LRM adopts high-intensity lighting in its dataset rendering process, making some objects blend into the white background. Since our model was not trained under such a light condition, it struggles with these cases and produces semi-transparent Gaussian predictions (e.g., the cap in Figure 1), harming the metric numbers.

Figure 1. **Failure Case on PF-LRM GSO Evaluation Dataset.**

2.2. Mesh Reconstruction Quality

Mesh is the most commonly used 3D representation in industry. Although our reconstruction model predicts Gaussians, meshes can be extracted by rendering multi-view depth maps and then executing TSDF fusion. We compare the quality of extracted meshes on the GSO evaluation set using FreeSplatter-O, and compare with pose-dependent LRMs. We adopt Chamfer Distance and F-Score (theshold 0.2) as the metrics, computed using 16K sampled points from both reconstructed and ground truth meshes. The results in Table 1 demonstrate FreeSplatter's superior performance on the OmniObject3D dataset across all metrics,

Figure 2. **Sparse-view Reconstruction on PF-LRM's Evaluation Datasets.** The first 2 rows are from the GSO evaluation set of PF-LRM, while the last 2 rows are from its OmniObject3D evaluation set. FreeSplatter-O synthesizes significantly better visual details than PF-LRM.

while achieving comparable 3D reconstruction quality with InstantMesh on the GSO dataset.

Method	OmniObject3D				
	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	F-Score@0.2 ↑
LGM*	24.852	0.942	0.060	0.073	0.766
InstantMesh*	24.077	0.945	0.062	0.044	0.882
FreeSplatter-O	31.929	0.973	0.027	0.043	0.896
	GSO				
	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	F-Score@0.2 ↑
LGM*	24.463	0.891	0.093	0.041	0.811
InstantMesh*	25.421	0.891	0.095	0.024	0.970
FreeSplatter-O	30.443	0.945	0.055	0.028	0.960

Table 1. **Mesh Reconstruction on GSO and OmniObject3D.** * indicates that ground truth camera poses are used as input.

2.3. Inference Efficiency

We conducted comprehensive benchmarking on a single A100 GPU, measuring both memory consumption and processing time (including Gaussian prediction and camera pose estimation). The results are in Table 2.

Input View Number	GPU memory (MB)	Inference time (s)
2	3611	1.268
3	3897	2.458
4	4225	3.669
5	4527	5.392
6	4837	7.495

Table 2. Time and memory cost during inference.

2.4. Results on Image-to-3D Generation

In this section, we provide extensive results on image-to-3D generation by combining FreeSplatter-O with different multi-view diffusion models.

Qualitative Results. In Figure 3 and Figure 4, we compare FreeSplatter's image-to-3D generation results with pose-dependent LRMs, *i.e.*, InstantMesh and LGM, using Zero123++ v1.2 [9] and a recent model Hunyuan3D Std [12] as the multi-view generator, respectively. For each input image, we fix the random seed to generate multi-view images, and visualize the novel view synthesis results of each reconstruction model. From the results, we can observe that FreeSplatter generates significantly more clear views and preserves the geometry and texture details better than other baselines.

In Figure 5, we show another interesting use case of FreeSplatter, i.e., using the input image to enhance the image-to-3D generation results. For multiview diffusion models like Zero123++ v1.1 [9], it generates 6 views from an input image at predefined poses, but *the pose of the input image is unknown* (its azimuth is known to be $0°$, but elevation is unknown). In this case, classical pose-dependent LRMs cannot leverage the input image for reconstruction, but FreeSplatter is able to do this! As Figure 5 shows, using the input image alongside generated views can significantly enhance the reconstruction results in many cases, especially for contents that Zero123++ struggles to generate, *e.g.*, human faces. Compared to generated views, the input image is often more high-quality and contains richer visual details. The pose-free nature of FreeSplatter makes it capable of exploiting these precious information in the reconstruction process.

In Figure 6, we show that FreeSplatter can faithfully recover the predefined camera poses of existing multi-view diffusion models from its reconstructed Gaussian maps. This demonstrates its robustness to generated multi-view images which may contain inconsistent contents.

Quantitative Results. Using the GSO evaluation dataset, we perform single-image-to-3D generation and evaluate the view synthesis and mesh extraction metrics (we use Zero123++ v1.2 for multi-view generation). The results in Table 3 demonstrate the significant performance advantages of FreeSplatter over pose-dependent baselines across all metrics.

Method	**GSO**				
	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	F-Score@0.2 ↑
LGM*	22.164	0.879	0.206	0.318	0.717
InstantMesh*	21.974	0.872	0.190	0.162	0.884
FreeSplatter-O	24.726	0.898	0.143	0.156	0.891

Table 3. **Single Image-to-3D Generation on GSO dataset.** * indicates that ground truth camera poses are used as input.

Figure 3. **Single Image-to-3D generation with Zero123++ v1.2 [9].**

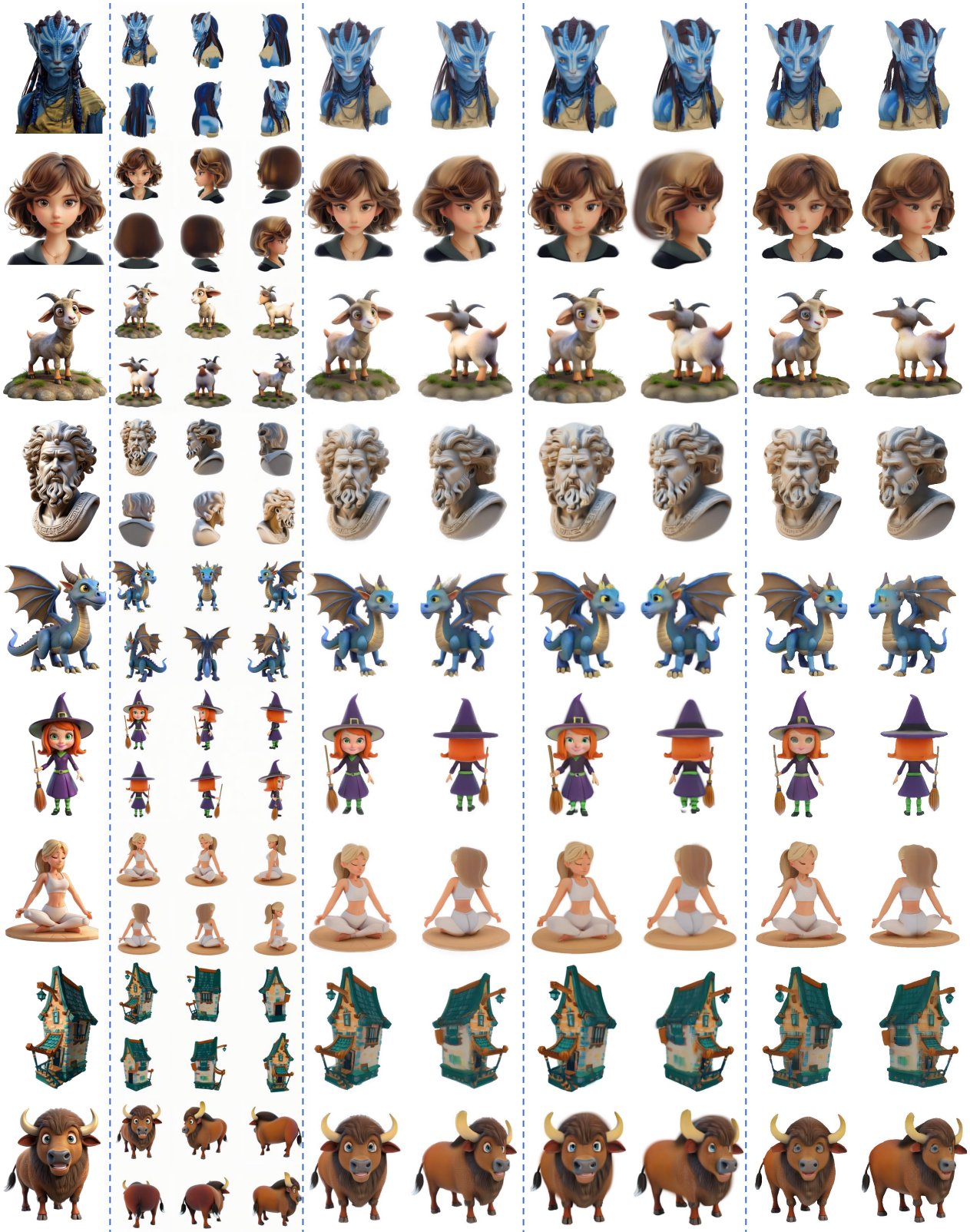

Figure 4. **Single Image-to-3D generation with Hunyuan3D Std [12].**

Figure 5. **Single Image-to-3D Generation with Zero123++ v1.1 [9].** FreeSplatter can use the input image alongside generated views to enhance the generation results, which is particularly valuable for challenging content like human faces, where Zero123++ often struggles to generate. The input image is often more high-quality and contains richer visual details than the generated views, but its camera pose is unknown, making it impossible for pose-dependent LRMs to leverage it.

Figure 6. **Camera Pose Estimation on Images Generated by Multi-view Diffusion Models.** FreeSplatter can faithfully recover the pre-defined camera poses of different multi-view diffusion models. We visualize the predefined camera poses of diffusion models and the estimated poses with gray and colorful pyramids, respectively. φ and θ denote the predefined azimuth and elevation angles, respectively. For Zero123++ v1.2 and Hunyuan3D Std which generate images at predefined fixed focal lengths, we mark the predicted focal lengths (in pixel units) too.

2.5. Comparison with NoPoSplat

NoPoSplat [13] also aims to predict 3D Gaussians from unposed images, sharing a similar motivation and high-level pipeline with FreeSplatter. However, there are significant differences between the two approaches: (i) NoPoSplat uses MASt3R's encoder-decoder architecture and requires weights initialization, FreeSplatter uses a pure ViT-based transformer and is trained from scratch; (ii) NoPoSplat assumes known camera intrinsics, whereas FreeSplatter requires no camera parameters; (iii) NoPoSplat focuses on scene-level images, while FreeSplatter is designed for both object-centric and scene-level reconstruction, offering broader applicability.

We compare FreeSplatter-S with the best NoPoSplat model jointly trained on Re10K and ACID. Table 4 and Figure 7 show that NoPoSplat performs better on Re10K view synthesis which we did not use for training, FreeSplatter out performs it on ScanNet++ and CO3Dv2. For pose estimation, our model consistently outperforms NoPoSplat.

Figure 7. **Comparison with NoPoSplat on Re10K (top) and ScanNet++ (bottom).**

Dataset	Method	PSNR ↑	SSIM ↑	LPIPS ↓	RRE ↓	RRA@15° ↑	RRA@30° ↑	TE↓
ScanNet++	NoPoSplat	21.477	0.782	0.255	1.765	0.971	0.979	0.483
	Ours	25.807	0.887	0.140	0.776	0.991	0.990	0.066
CO3Dv2	NoPoSplat	17.380	0.626	0.288	5.296	0.921	0.938	0.327
	Ours	20.405	0.781	0.162	3.048	0.976	0.986	0.190
Re10K	NoPoSplat	23.867	0.819	0.212	3.467	0.966	0.991	0.397
	Ours	20.513	0.746	0.350	3.513	0.982	0.995	0.293

Table 4. **Comparison with NoPoSplat.**

2.6. Cross-dataset Generalization

Object-centric Datasets. To demonstrate the cross-dataset generalization ability of FreeSplatter-O, we visualize the object-centric reconstruction results on OmniObject3D and ABO [2] datasets in Figure 8. Both datasets are outside the training scope of FreeSplatter-O. We can observe that FreeSplatter-O generates very high-quality renderings.

Scene-level Datasets. We test FreeSplatter-S on RealEstate10K [16], which we did not use for training. The results are visualized in Figure 9, showing that our model can faithfully reconstruct the input views at the estimated input poses, while the rendered novel views align well with the ground truth. We further demonstrate FreeSplatter-S's broad generalization capabilities through extensive evaluation across diverse datasets in Figure 10, covering scanned objects (DTU [3]), large-scale outdoor scenes (Tanks & Temples [4]), multi-view object collections (MVImgNet [14]), and AI-generated synthetic content (videos generated by Sora[2]). These results validate our framework's robustness across varying scene types and capture conditions.

[2] https://openai.com/index/sora/

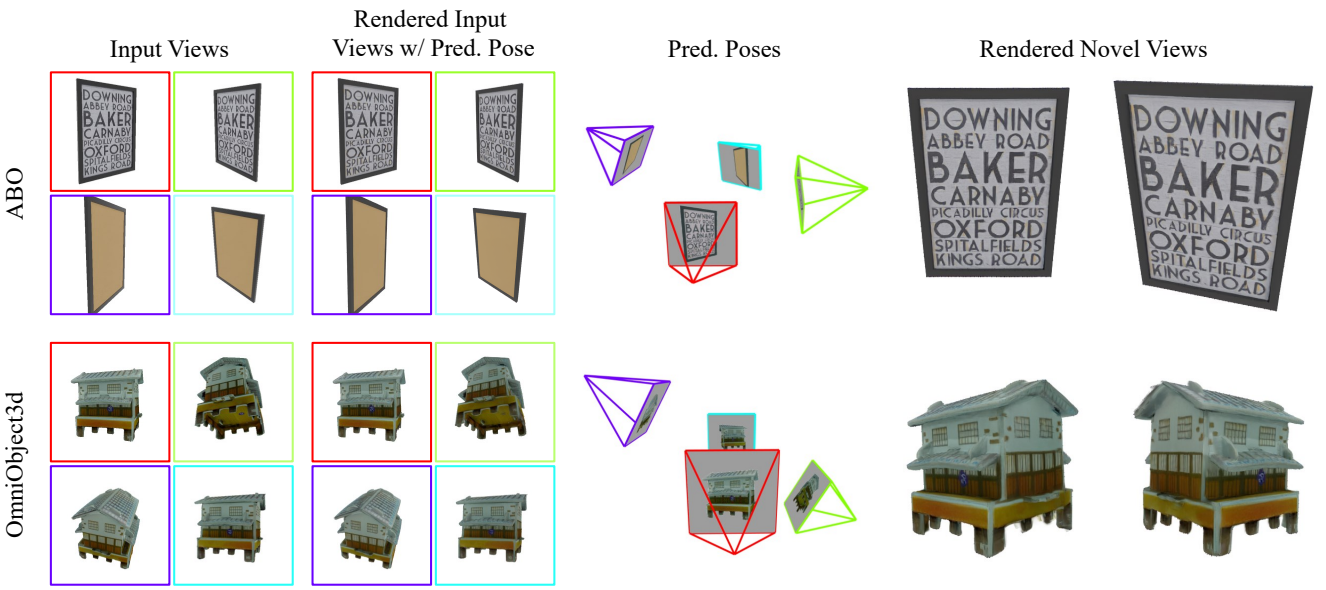

Figure 8. **Sparse-view Reconstruction and Camera Pose Estimation on ABO and OmniObject3D.** Both datasets are unseen for FreeSplatter-O. The model can faithfully estimate the input camera poses and render high-fidelity novel views.

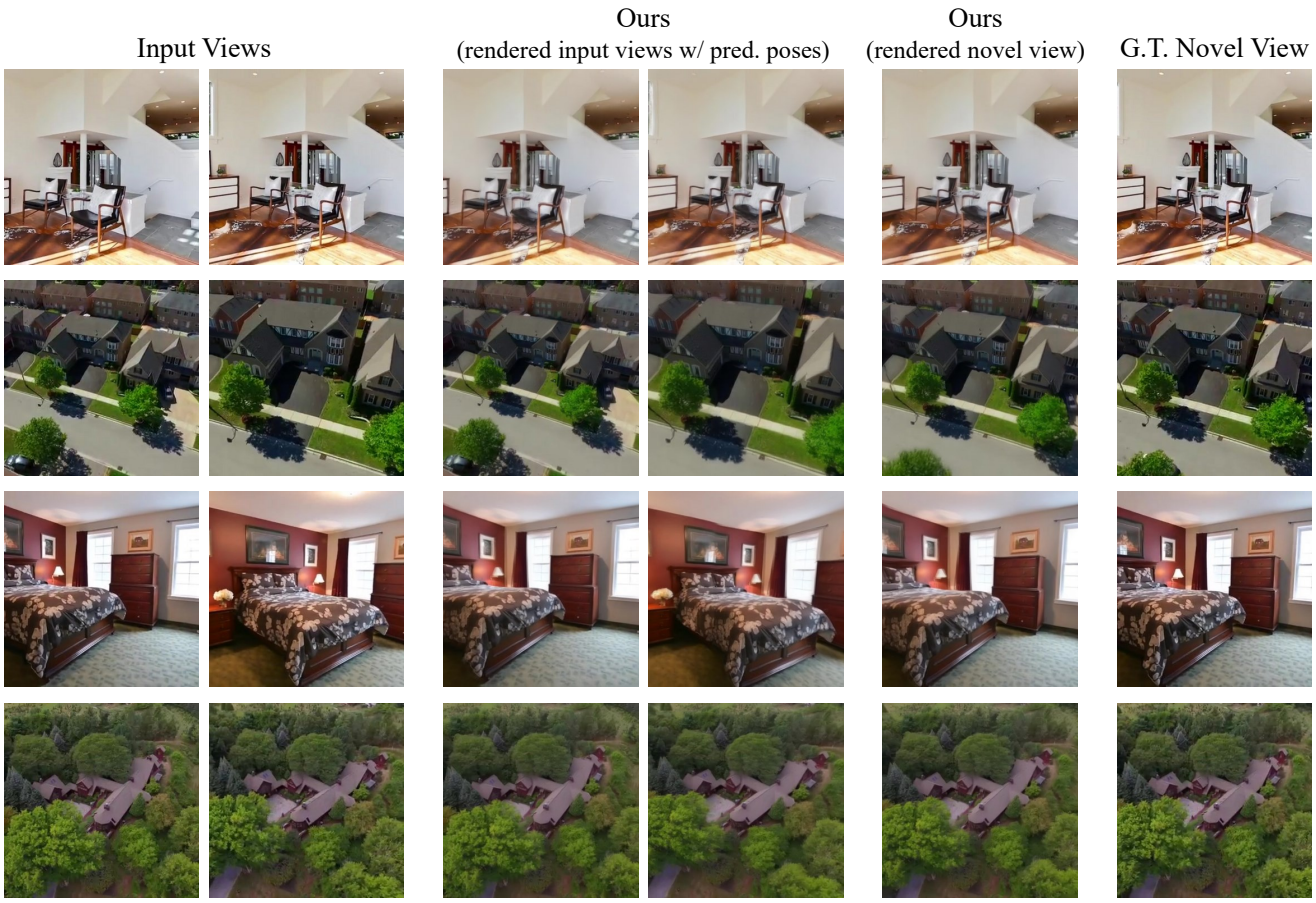

Figure 9. **Sparse-view Reconstruction on RealEstate10K.** Our FreeSplatter-S model was not trained on RealEstate10K, we directly utilize it for zero-shot view synthesis on this dataset. We can observe that our model can faithfully reconstruct the input views at the estimated input poses, and the rendered novel views align well with the ground truth.

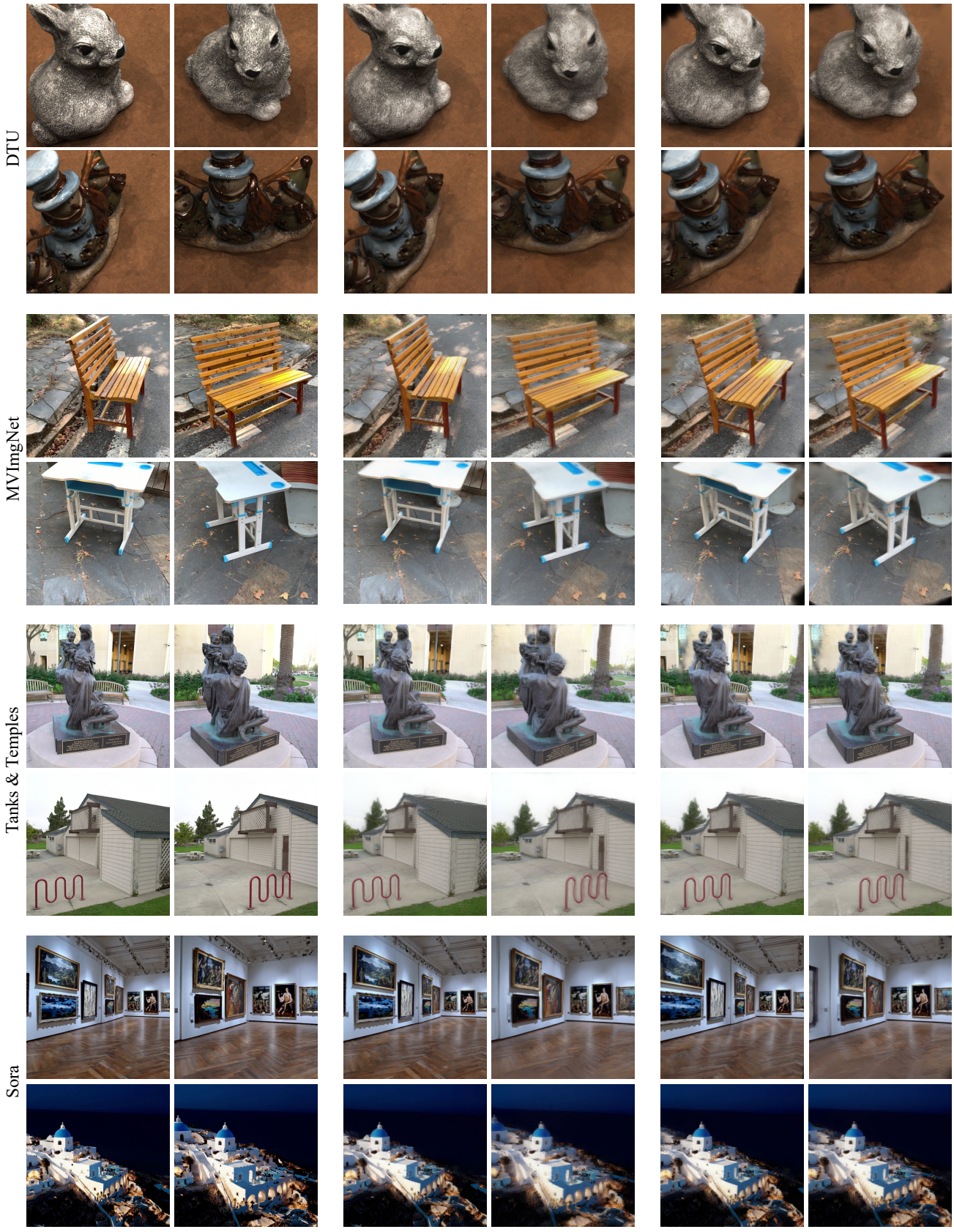

Figure 10. **Sparse-view Reconstruction on Various Scene-level Datasets.** We show 2 examples for DTU, MVImgNet, Tanks & Temples and Sora-generated videos, respectively.

2.7. Ablation Studies

Due to the page limit of the main manuscript, we have put additional results of ablation studies here.

View Embedding Addition. We evaluate the effectiveness of adding view embeddings to image tokens as in Equation 2. In FreeSplatter, we add the multi-view image tokens with a reference view embedding $\mathbf{e}_{\text{ref}}$ or a source view embedding $\mathbf{e}_{\text{src}}$ before feeding them into the transformer to make the model identify the reference view, so that it can reconstruct Gaussians in the reference view's camera frame. We present the results in Figure 11.

Specifically, for 4 input views V_i $(i = 1, 2, 3, 4)$, we try different combinations of $\mathbf{e}_{\text{ref}}$ and $\mathbf{e}_{\text{src}}$ when adding them to the image tokens, and then render the reconstructed Gaussians with an "identity camera pose" $C = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]$ to see the results. We can observe that, when adding the j-th view's tokens with $\mathbf{e}_{\text{ref}}$ and other views' tokens with $\mathbf{e}_{\text{src}}$, the rendered image is exactly the j-th view. This means that the model successfully identify the j-th view as the reference view and reconstruct Gaussians in its camera frame. In comparison, all other view embedding combinations leads to degraded reconstruction.

Number of Input Views. We make an experiment on a GSO sample to illustrate how the number of input views influences the reconstruction quality on the object-centric scenarios. Please refer to Figure 13 in the appendix for more details.

Pixel-Alignment Loss. We try removing the pixel-alignment loss defined in Equation 5 (main paper) from the training of both models, and report the novel view rendering metrics on GSO and ScanNet++ dataset. The significant drops in all metrics indicates its importance. The visual results in Figure 12 show that the model trained without pixel-alignment loss produces blurry renderings, while adopting it leads to significantly better visual details.

Position Loss. As noted in Section 3.3, the position loss defined in Equation 4 is essential-without it, the model diverges. We ablate its effectiveness in Table 5, from which we can observe that the performance drops sharply on GSO without $\mathcal{L}_{\text{pos}}$.

$\mathcal{L}_{\text{pos}}$	$\mathcal{L}_{\text{align}}$	PSNR↑	SSIM↑	LPIPS↓
✗	✗	10.510	0.476	0.327
✗	✓	11.218	0.494	0.306
✓	✗	26.684	0.898	0.092
✓	✓	30.443	0.945	0.055

Table 5. **Ablation Study on Position Loss.** The results are evaluated on the GSO dataset with FreeSplatter-O.

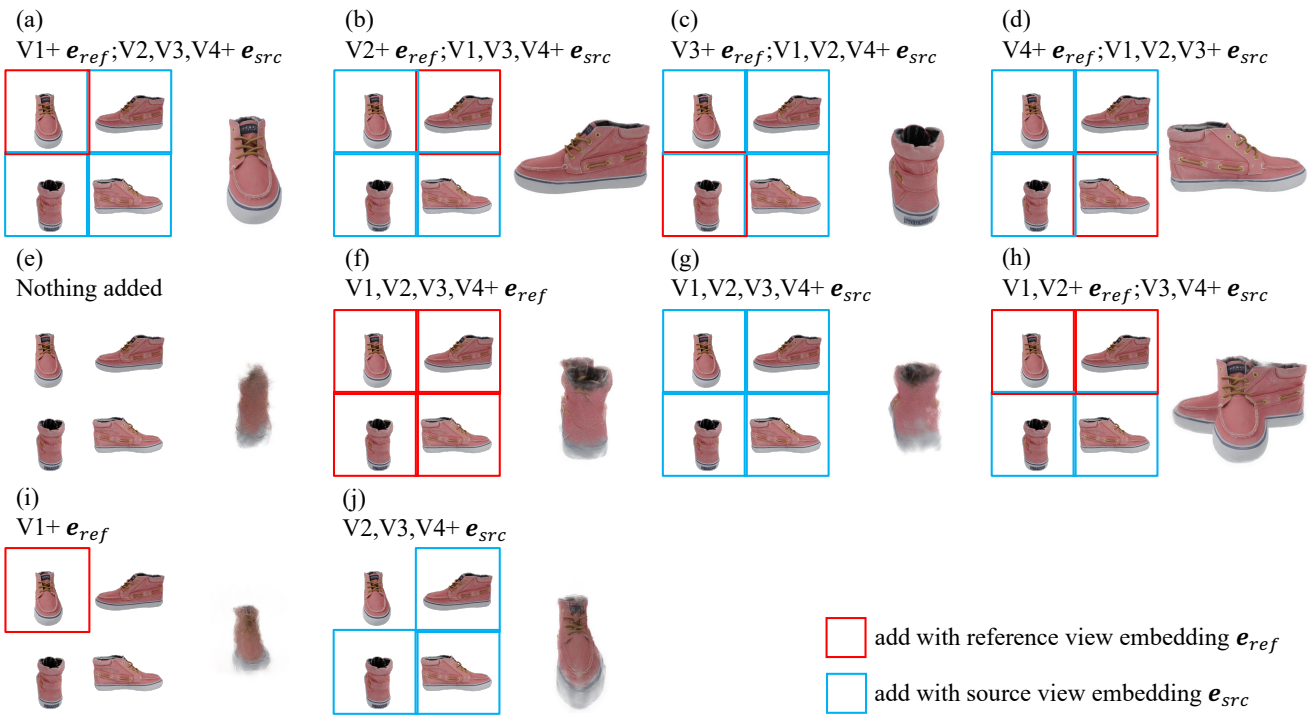

Figure 11. **Ablation Study on View Embedding Addition.** Red/blue boxes indicate views added with reference/source view embeddings respectively. For each case, we visualize the image rendered with identity camera (*i.e., reference pose*) on the right.

Figure 12. **Ablation Study on Pixel-alignment Loss.** We show two samples from the GSO dataset.

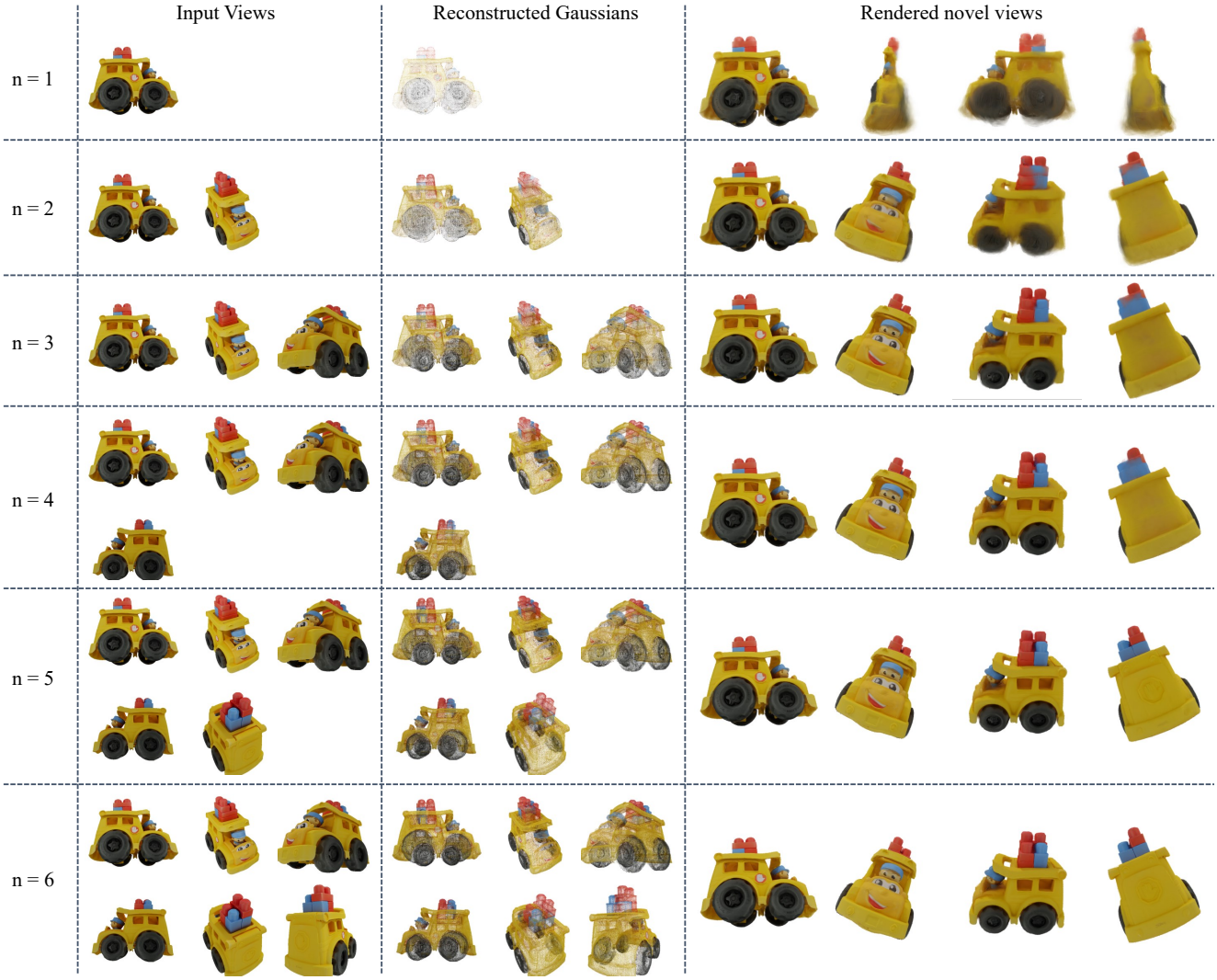

Figure 13. **Ablation Study on Number of Input Views.** We show the visual comparison of FreeSplatter-O results with varying numbers of input views ($n = 1 - 6$). From left to right: input views, reconstructed Gaussians, and rendered target views at 4 fixed viewpoints. Additional input views increase Gaussian density and improve previously uncovered regions, with diminishing returns beyond n=4 when object coverage becomes sufficient.

References

[1] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 1

[2] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022. 9

[3] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 9

[4] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 9

[5] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022. 1

[6] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[7] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 1

[8] Frank Plastria. The weiszfeld algorithm: proof, amendments, and extensions. *Foundations of location analysis*, pages 357–389, 2011. 1

[9] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 4, 5, 7

[10] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. PF-LRM: Pose-free large reconstruction model for joint pose and shape prediction. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[11] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 1, 2

[12] Xianghui Yang, Huiwen Shi, Bowen Zhang, Fan Yang, Jiacheng Wang, Hongxu Zhao, Xinhai Liu, Xinzhou Wang, Qingxiang Lin, Jiaao Yu, et al. Hunyuan3d-1.0: A unified framework for text-to-3d and image-to-3d generation. *arXiv preprint arXiv:2411.02293*, 2024. 4, 6

[13] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. 9

[14] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023. 9

[15] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields, 2022. 1

[16] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph*, 37, 2018. 9