Generative Densification: Learning to Densify Gaussians for High-Fidelity Generalizable 3D Reconstruction

Supplementary Material

A. Additional Results on the DL3DV Dataset

Table 1. Evaluation results on the DL3DV-10k [8] dataset. n denotes the frame distance span across all test views

Method	$n = 150$			$n = 300$		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat-finetune	17.42	0.516	0.417	16.19	0.452	0.478
Ours	**17.76**	**0.533**	**0.405**	**16.46**	**0.468**	**0.469**

We further evaluate our scene-level model on the DL3DV-10k [8], a challenging dataset with 51.3 million frames from 10,510 real-world scenes. Our scene-level model and its baseline are fine-tuned for 100,000 iterations on two subsets ("3k" and "4k") of the DL3DV-10k dataset, comprising approximately 2,000 scenes. All models are evaluated on 140 scenes that are filtered out from the training set, following MVSplat360 [2]. For each scene, we select 5 views as input and evaluate on 56 views uniformly sampled from the remaining views [2]. As shown in Tab. 1, our model outperforms the baseline across all metrics, consistent with the evaluations on the RE10k, ACID, and DTU datasets.

B. Generating Residuals of Fine Gaussians

We further propose to generate residuals of fine Gaussians for scene-level reconstruction. Similar to the densification procedure presented in Sec. 3.1, the top K Gaussians with large view-space positional gradients ($\mathcal{G}_{\text{den}}$) are selected, and their positions and features ($\mathcal{X}_{\text{den}}$, $\mathcal{F}_{\text{den}}$) are passed through L layers of serialized attention (SA), up-sampling (UP), Gaussian head (HEAD), and splitting (SPLIT) blocks:

$$(\tilde{\mathcal{X}}^{(l)}, \tilde{\mathcal{F}}^{(l)}) = \texttt{SA}(\mathcal{X}_{\text{den}}^{(l-1)}, \mathcal{F}_{\text{den}}^{(l-1)}), \quad (1)$$

$$(\mathcal{X}^{(l)}, \mathcal{F}^{(l)}) = \texttt{UP}(\tilde{\mathcal{X}}^{(l)}, \tilde{\mathcal{F}}^{(l)}), \quad (2)$$

$$\mathcal{G}^{(l)} = \texttt{HEAD}(\mathcal{G}_{\text{den}}^{(l-1)}, \mathcal{X}^{(l)}, \mathcal{F}^{(l)}), \quad (3)$$

$$(\mathcal{G}_{\text{den}}^{(l)}, \mathcal{X}_{\text{den}}^{(l)}, \mathcal{F}_{\text{den}}^{(l)}, \mathcal{G}_{\text{rem}}^{(l)}) = \texttt{SPLIT}(\mathcal{G}^{(l)}, \mathcal{X}^{(l)}, \mathcal{F}^{(l)}), \quad (4)$$

for $l \in \{1, \cdots, L-1\}$, where $\mathcal{G}_{\text{den}}^{(0)} = \mathcal{G}_{\text{den}}$, $\mathcal{X}_{\text{den}}^{(0)} = \mathcal{X}_{\text{den}}$, and $\mathcal{F}_{\text{den}}^{(0)} = \mathcal{F}_{\text{den}}$. The Gaussian positions and features are up-sampled in the UP block, which is identical to the up-sampling procedure as described in Sec. 3.2. The up-sampled positions and features are then passed to the HEAD block, where they are transformed into Gaussian parameters and added to those from the previous layer ($\mathcal{G}_{\text{den}}^{(l-1)}$). In the SPLIT block, the generated Gaussians are divided into two groups: those that require further densification and those that do not. The first group is refined in the next layer, while the second group remains unchanged. Note that the second group of Gaussians' positions and features ($\mathcal{X}_{\text{rem}}^{(l)}, \mathcal{F}_{\text{rem}}^{(l)}$) are omitted from Eq. (4) for brevity. Finally, the L-th layer's fine Gaussians are generated as $\mathcal{G}^{(L)} = \texttt{HEAD}(\mathcal{G}_{\text{den}}^{(L-1)}, \texttt{UP}(\texttt{SA}(\mathcal{X}_{\text{den}}^{(L-1)}, \mathcal{F}_{\text{den}}^{(L-1)})))$ without splitting, and the final set of Gaussians is obtained as:

$$\hat{\mathcal{G}} = \{\bigcup_{l=0}^{L-1} \mathcal{G}_{\text{rem}}^{(l)}\} \cup \mathcal{G}^{(L)}. \quad (5)$$

The two densification methods for object-level and scene-level reconstruction are similar in that both selectively generates fine Gaussians leveraging learnable masking, but they differ in how fine Gaussians are generated. The object-level method generates fine Gaussians directly in each densification layer, while the scene-level method generates initial fine Gaussians in the first layer and selectively refines them by adding residuals in subsequent layers.

C. Model Details

The Number of Gaussians. As described in Sec. 3.1, we begin by selecting $K^{(0)} = K$ Gaussians, which are densified through L layers of densification blocks. In the l-th up-sampling block, both Gaussian positions and features are up-sampled by a factor of $R^{(l)}$, resulting in an increased Gaussian count, $K^{(l)} = K^{(l-1)}R^{(l)}$. Subsequently, in the splitting module, the up-sampled Gaussians are divided into two groups: those requiring further densification in the next layer, and those that do not. The number of Gaussians for further densification, $K_{\text{den}}^{(l)}$, and the remaining Gaussians, $K_{\text{rem}}^{(l)}$, are calculated as $K_{\text{den}}^{(l)} = \lceil K^{(l)}P^{(l)} \rceil$ and $K_{\text{rem}}^{(l)} = K^{(l)} - K_{\text{den}}^{(l)}$, respectively, where $P^{(l)} \in (0, 1)$ is the masking ratio and $\lceil \cdot \rceil$ represents the ceiling operator.

Global Adaptive Normalization. The serialized attention module is learned to aggregate the scene context but operates within each group of Gaussians for memory and computational efficiency, which may lead to limited understanding of the global context. To complement the local features, a global feature is widely used as a global descriptor in point-level architectures. Inspired by previous works [12, 13] and recent normalization techniques [11, 15], we introduce global adaptive normalization, which averages the features of the Gaussians selected for densifcation ($\mathcal{F}_{\text{den}}$) and scales the normalized features using the averaged features.

Generative Densification of LaRa. LaRa [1] generates 3D volume representations conditioned on image features,

Table 2. Summary of training hyperparameters.

Object-level		Scene-level	
Config	Value	Config	Value
optimizer	AdamW [10]	optimizer	Adam [7]
scheduler	Cosine	scheduler	Cosine
learning rate	4e-4	learning rate	2e-4
beta	[0.9, 0.95]	beta	[0.9, 0.999]
weight decay	0.05	weight decay	0.00
warmup iters	1,000	warmup iters	2,000
epochs	30	iters	300,000

Table 3. Summary of fine-tuning hyperparameters.

Object-level		Scene-level	
Config	Value	Config	Value
optimizer	AdamW [10]	optimizer	Adam [7]
scheduler	Cosine	scheduler	Cosine
learning rate	2e-4	learning rate	2e-4
beta	[0.9, 0.95]	beta	[0.9, 0.999]
weight decay	0.05	weight decay	0.00
warmup iters	0	warmup iters	0
epochs	20	iters	150,000

and each volume features are decoded into multiple Gaussians. To improve the rendering quality, LaRa introduce a cross-attention between the volume features and coarse renderings, including ground-truth images, rendered images, depth images, and accumulated alpha maps [1]. The intermediate features from the cross-attention are transformed to residuals of SH using an MLP, which are added to the coarse SH to obtain the refined Gaussians.

We modify the last MLP (the residual SH decoder) to output both the residuals and refined volume features. The first d columns of the MLP output are considered the residual SH, while the remaining columns serve as input features for generative densification. Here, d denotes the number of SH coefficients. We take the refined Gaussians and the concatenation of volume and refined volume features as input, and their respective fine Gaussians are generated by our method. Note that, while the baseline LaRa generates 2D Gaussian representations [5], we adapt it to generate 3D Gaussians representations [6] instead.

Generative Densification of MVSplat. MVSplat [3] generates per-view pixel-aligned Gaussian representations from multi-view input images. A transformer encodes the input images into features via cross-view attention, after which per-view cost volumes are constructed. The image features are concatenated with these cost volumes and decoded into depths and other parameters, including opacities, covariances, and colors. The Gaussian positions are then determined by un-projecting the depths into 3D space.

Similar to LaRa, we obtain the refined features by applying cross-attention between the coarse renderings and the concatenated features of images and cost volumes, followed by a simple MLP. However, we do not predict residuals of SH, as this often leads to unstable training in scene-level reconstruction. We use the per-view Gaussians from the MVSplat backbone along with the refined features as input to our method, generating per-view fine Gaussians.

Impelmentation Details. For object-level reconstruction, we select 12,000 Gaussians from the LaRa backbone with large view-space gradients and generate fine Gaussians through two densification layers. The up-sampling factors of the two layers are 2 and 4, and the masking ratio is 0.8. In other words, the input Gaussians are densified by a factor of 2 in the first layer, and 80% of them are further densified by a factor of 4 in the second layer, while the remaining 20% are decoded into raw Gaussians. Similarly, for scene-level reconstruction, we select 30,000 Gaussians per view from the MVSplat backbone. We use three densification layers, each with an up-sampling factor of 2. The masking ratios are set to 0.5 for the first layer and 0.8 for the second layer.

D. Training and Evaluation Details

The training and fine-tuning hyperparameters are summarized in Tab. 2 and Tab. 3, respectively. The training objectives and additional details are outlined in the followings.

Object-level Reconstruction. The backbone and the densification module are jointly trained by minimizing the loss $\mathcal{L} = \mathcal{L}_{\text{MSE}}(\mathcal{I}, \hat{\mathcal{I}}) + 0.5(1 - \mathcal{L}_{\text{SSIM}}(\mathcal{I}, \hat{\mathcal{I}}))$, for both coarse and fine images, where $\mathcal{L}_{\text{MSE}}$ is the mean squared error and $\mathcal{L}_{\text{SSIM}}(\mathcal{I}, \hat{\mathcal{I}})$ is the structural similarity loss. The coarse images are rendered using the Gaussians generated by the backbone LaRa, and the fine images are rendered using the densified fine Gaussians. Unlike the original implementation [1], where the fine decoder (the last cross-attention layer and the residual SH decoder) is trained after the first 5,000 iterations, we train it from the very beginning.

For model evaluation on the GSO dataset, we utilize the classical K-means algorithm to group the cameras into 4 clusters and select the center of each cluster to ensure sufficient angular coverage of the input views. Both LaRa and our model are evaluated using this new sampling method.

Scene-level Reconstruction. Similar to our object-level model, we calculate the image reconstruction loss $\mathcal{L} = L_{\text{MSE}}(\mathcal{I}, \hat{\mathcal{I}}) + 0.05\mathcal{L}_{\text{LPIPS}}(\mathcal{I}, \hat{\mathcal{I}})$ for both coarse and fine images, and minimize the loss to jointly train the MVSplat backbone and the densification module. Here, $\mathcal{L}_{\text{LPIPS}}$ denotes the learned perceptual similarity loss (LPIPS [16]). For model fine-tuning, we set the warm-up step of the view-sampler to 0, and the minimum and maximum distances between context views are set to 45 and 192, respectively.

E. Additional Qualitative Results

Fig. 1 illustrates how fine Gaussians are generated in each densification layer. Fig. 2 and Fig. 3 show comparisons for object-level and scene-level reconstruction, respectively. Fig. 4 show rendering results from far-away novel views.

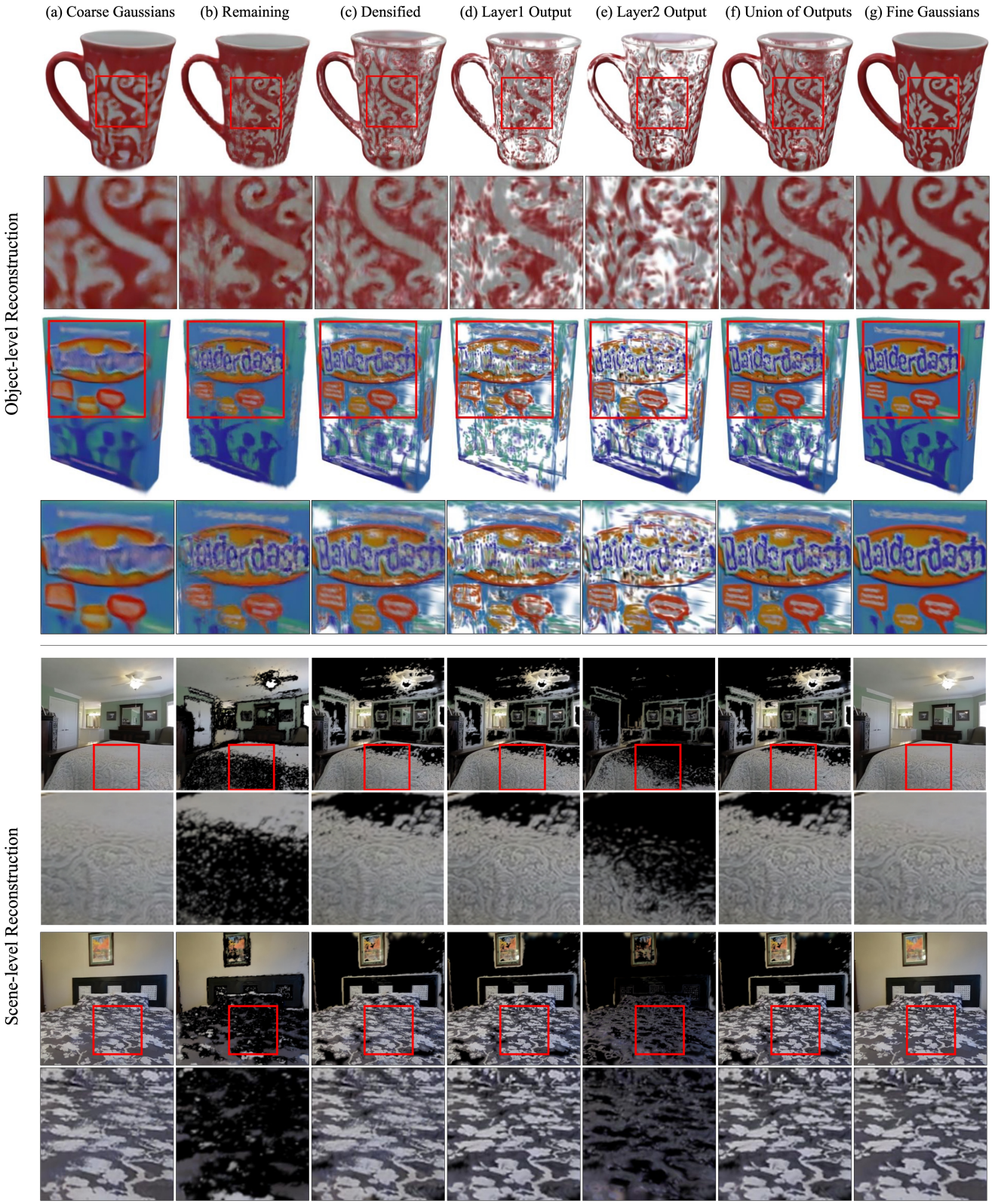

Figure 1. Additional qualitative results of our object-level and scene-level model trained for 50 epochs and 450,000 iterations, respectively. The zoomed-in parts show our method selects and reconstructs the fine details through alternating densification layers, while preserving the smooth areas unchanged. Note that the 7-th column of the scene-level reconstruction results shows the union of fine Gaussians generated across all three densification layers, and the output Gaussians from the third layer are omitted due to space constraints.

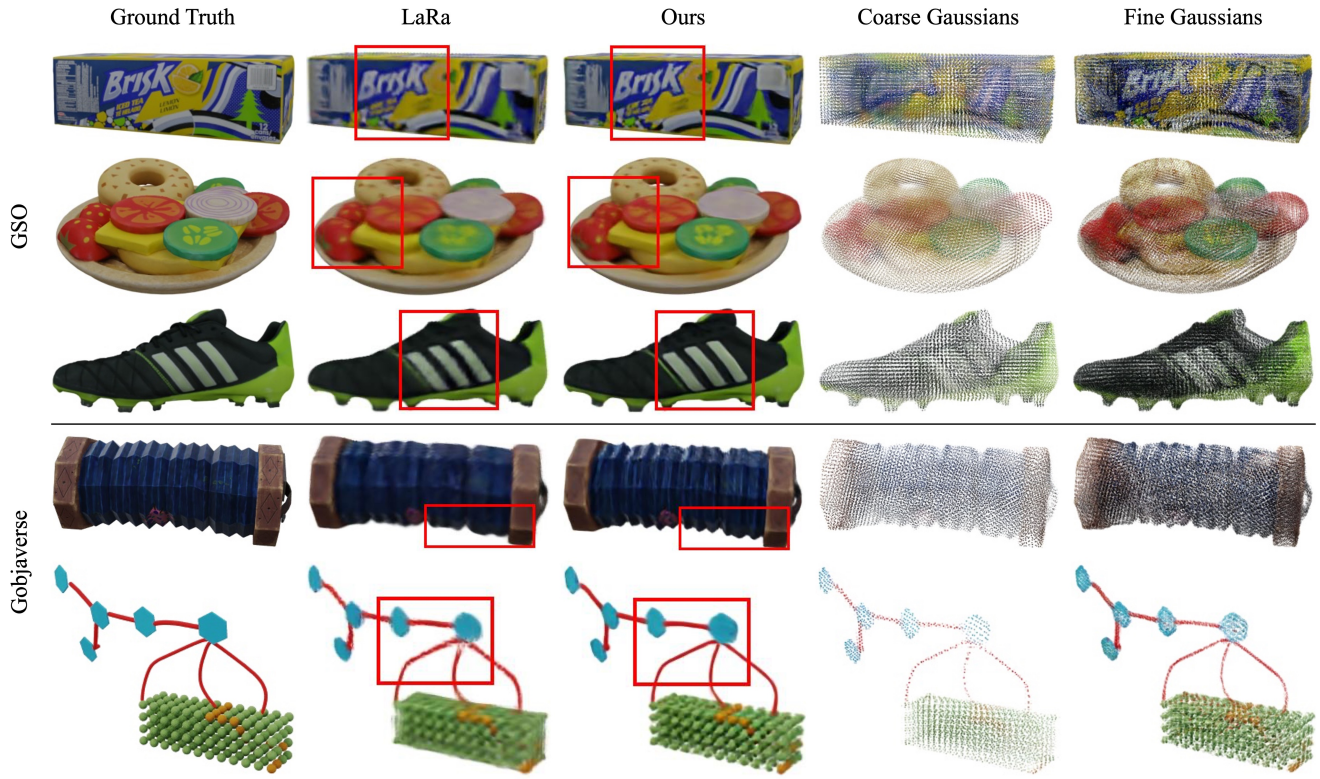

Figure 2. Qualitative comparisons of our object-level model against the original LaRa [1], evaluated on the GSO [4] and Gobjaverse [14] dataset. The coarse and fine Gaussians are the input and output of generative densification module, respectively.

Figure 3. Qualitative comparisons of our scene-level model against the original MVSplat [3], evaluated on the RE10K [17] dataset. The red boxes show that our model better reconstructs the scene, removing visual artifacts and generating missing parts.

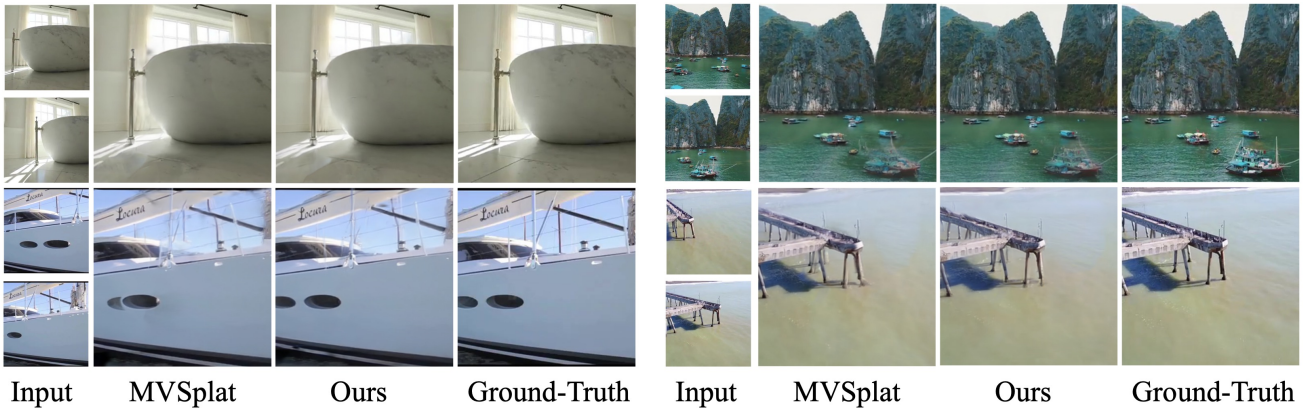

Figure 4. Qualitative comparisons of our scene-level model against the original MVSplat [3], evaluated on the RE10K [17] dataset (left column) and the ACID [9] dataset (right column). Our method consistently improves reconstruction quality at far away from input views.

References

[1] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *European Conference on Computer Vision*, 2025. 1, 2, 4

[2] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. *arXiv preprint arXiv:2411.04924*, 2024. 1

[3] 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*, 2025. 2, 4, 5

[4] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *International Conference on Robotics and Automation*, 2022. 4

[5] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 2

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 2

[7] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[8] 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*, 2024. 1

[9] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 5

[10] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[11] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1

[12] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. 1

[13] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 1

[14] Chao Xu, Yuan Dong, Qi Zuo, Junfei Zhang, Xiaodan Ye, Wenbo Geng, Yuxiang Zhang, Xiaodong Gu, Lingteng Qui, Zhengyi Zhao, Qing Ran, Jiayi Jiang, Zilong Dong, and Liefeng Bo. G-buffer objaverse: High-quality rendering dataset of objaverse. https://aigc3d.github.io/gobjaverse/. 4

[15] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in Neural Information Processing Systems*, 2019. 1

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[17] 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. 4, 5