

# NVGS: Neural Visibility for Occlusion Culling in 3D Gaussian Splatting

## Supplementary Material

### A. Supplementary

This supplementary document provides additional results and evaluations for all layout files, including renders and FLIP [1] comparisons for each layout. This includes comparisons for the full layouts from V3DG [28], where we use our instanced rasterizer without the MLP for reference renderings. Results for individual assets are presented, along with visual comparisons of the ground-truth contributions and our predicted values. We also encourage the reader to view the supplementary videos, as the results can be more clearly inspected when the camera is in motion.

#### A.1. Per-asset Results

In Tab. 5 we show per-asset results. These metrics were measured for a circular camera trajectory around each asset. For the assets in the DONUTSEA and CROWD layouts, our MLP culls a significant number of Gaussians, ranging from  $\sim 29\%$  to  $\sim 40\%$ . For the trees from the FOREST layout, the number of Gaussians that can be pruned depends strongly on the crown structure and trunk thickness. In the cases of oak and dinosaur, we show in the video that these exact factors complicate the accurate learning of the visibility function. On average, our approach increases the percentage of useful Gaussians passed to the rasterizer while significantly decreasing the total number of Gaussians processed. Notably, these Gaussians can be discarded without loss of quality, as they do not contribute to the ground truth renders.

In Figs. 9 to 11, we present side-by-side images of a selection of assets, comparing the ground truth selection with the occlusions our MLP predicts. From left to right, we show the ground truth in the single view, the ground truth we actually learn using a small set of auxiliary views to counter training on popping artifacts, and what our MLP predicts as occluded. In all views, (predicted) non-contributing splats are rendered in red, as if the camera were on the opposite side of the asset. For the avatars and trees, we display the asset with the best and worst performance in terms of the percentage of Gaussians pruned. For the trees, notice that the two that perform worse have many small gaps in their crowns, resulting in a higher frequency of visibility changes compared to the avatars or donut. Including frequency encodings and avoiding auxiliary views both increase the number of Gaussians we predict as occluded, at the cost of rendering speed and robustness against popping artifacts due to harder-to-learn visibility functions. Depending on the application, the selection of process steps can be fine-tuned; however, our proposed method, with all steps enabled, achieves consistent results across assets and datasets.

#### A.2. Large-scale Scenes

Our approach enables the rendering of large, composed scenes with significantly less VRAM than other methods, due to the combination of our instanced rasterizer and occlusion-culling MLP. In Fig. 14, we show the renders and FLIP metrics on the downscaled scenes compared to V3DG [28] for the scenes not in the main paper, and in Fig. 15, we show the renders of the scene at full scale compared to our baseline. For both our method and V3DG, we observe a concentration of errors on the borders in the CROWD scene. While we the exact training setup of Yang *et al.* [28] is not publicly available, we speculate that this may

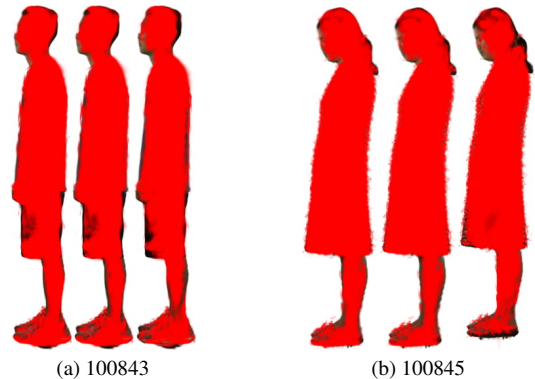


Figure 9. From left to right, we show the ground truth zero-contribution Gaussians, ground truth zero-contribution using 6 auxiliary views, and what the MLP predicts. The two chosen assets are the best- and worst-performing assets from the MVHuman-Net [27] dataset in terms of Gaussian pruning percentage.

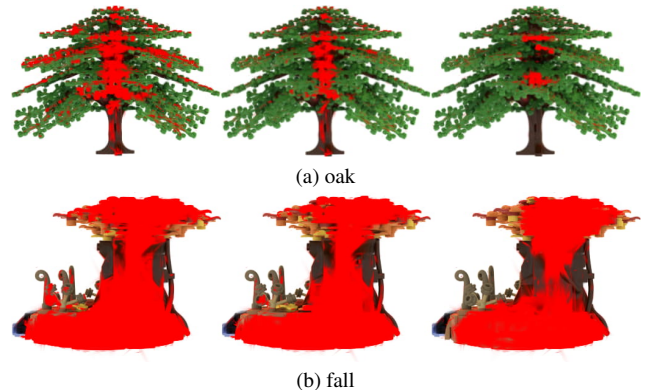


Figure 10. From left to right, we show the ground truth zero-contribution Gaussians, ground truth zero-contribution using 6 auxiliary views, and what the MLP predicts. The two chosen assets are the best- and worst-performing assets from the RTMV [24] trees subset in terms of Gaussian pruning percentage.

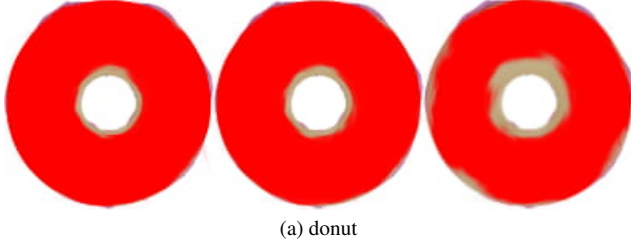


Figure 11. From left to right, we show the ground truth zero-contribution Gaussians, ground truth zero-contribution using 6 auxiliary views, and what the MLP predicts. We show this for the donut asset [25].

be due to their training on a background other than white. If this is the case, it can be solved by training on a random background, improving the results of both methods. Our approach can run these large scenes on an NVIDIA RTX 3090 Ti with only 24 GB of VRAM at >20 FPS.

### A.3. Characterization of Impact Factors

The impact of our approach on a given scene is influenced by several factors, such as the SH degree shown in Fig. 8. We include this discussion to make these factors explicit and thus clarifying the strengths and limitations of our approach.

**Compute per Gaussian** Our approach scales effectively as the computational complexity per Gaussian increases. This is a direct consequence of skipping the preprocessing of pruned Gaussians, which becomes more beneficial as per-Gaussian costs rise. In Fig. 8, we demonstrate this by increasing the degree of Spherical Harmonics (SH). For future work involving more complex per-Gaussian computations, *e.g.*, expensive shading/lighting routines, our approach provides a direct path for maintaining performance.

**Gaussian/Asset Density** As more Gaussians project to the same pixel, the standard tiled 3DGS rasterization algorithm becomes unable to balance loads effectively across tiles, which leads to reduced frame times. NVGS helps in mitigating this by reducing the number of Gaussians that require processing. While standard 3DGS was originally designed for spread distributions of Gaussians, the relative advantage of our approach grows in scenes with high Gaussian density, where assets are closely packed, and depth complexity per asset is high.

**Screen Coverage** Pruning Gaussians directly reduces the total number of generated Gaussian/tile instances. Large Gaussians that overlap several tiles are the most impactful candidates for pruning, as a single visibility query can prevent redundant instancing across multiple tiles. Conversely,

for small Gaussians contained within a single tile, the relative computational savings from using NVGS are lower.

**Occlusion Structure** Since we do not frequency-encode the MLP inputs (which would help it to learn high-frequency visibility functions [23]) to reduce computational demands, the occlusion structure of a scene influences performance. To address this, we simplify the visibility function by smoothing it through auxiliary views and antialiasing. While these simplifications aid convergence, they increase the number of Gaussians contributing to the final render, thereby reducing the number that can be pruned. Consequently, scene parts with high-frequency visibility functions, such as a tree’s crown, remain challenging for our method, as these simplifications increase the number of contributing Gaussians.

**Distance to the Object** The distance to the object also impacts performance. As our approach attempts to learn the asset’s visibility as is, we are limited in pruning by how many contribute at far distances. Dedicated LoD approaches, however, can use fewer Gaussians to represent far distances. Compared to such methods (*e.g.*, V3DG [28]), our methods perform well at close and medium distances, while they outperform us for far distances. For long distances, where these methods can use fewer Gaussians by merging them, we do not merge Gaussians. The biggest jump is thus achieved up-close as can be seen in Fig. 6.

### A.4. More Detailed Objects

To evaluate scalability of our approach w.r.t. to asset complexity, we utilize the temple asset provided by Yang *et al.* [28] and a downscaled version of their TEMPLES layout. The asset consists of approximately 1.3M Gaussians,

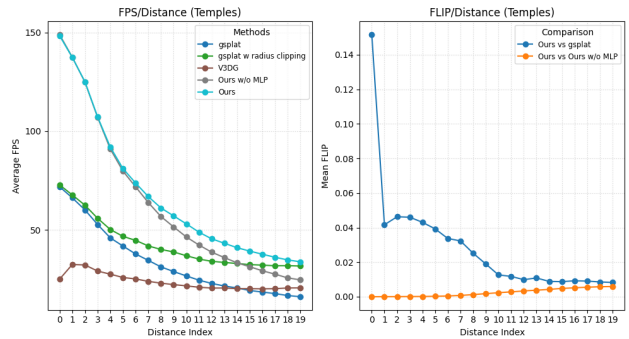


Figure 12. Average FPS/Distance plot for the TEMPLES layout file. We additionally show the FLIP comparison against both the reference (gsplat) and ours w/o MLP. Our approach consistently achieves higher FPS, but due to floaters (Fig. 13), the metrics compared to the gsplat reference are surprisingly low. The FLIP comparison against ours w/o MLP shows that this is not a failure of our method but a difference in the rasterizer.

which reduces to roughly 900k after standard opacity pruning ( $< \frac{1}{255}$ ). As shown in Fig. 12 and Tab. 4, our approach consistently maintains a lower VRAM footprint and higher FPS at close-to-medium distances. While image metrics for this scene are lower when compared to the reference rendering from gsplat, visual inspection reveals that this is primarily due our rasterizer being able to cull large “floaters” present in the original asset more effectively (see Fig. 13). This results in a better viewing experience and visual quality but does reduce image metrics. We highlight that, as clearly shown in Fig. 12, this behavior is independent of the visibility MLP predictions and solely depends on how the underlying rasterizer handles large splats.

Table 4. We show image metrics and VRAM usage for the TEMPLES layout from V3DG. Our approach uses less VRAM and higher FPS for all distances in the layout. For this particular layout, we notice that image metrics are low at first glance. We show in Fig. 13 that, although our approach visually appears the same, our rasterizer culling large Gaussians artificially deflates image metrics.

Method	PSNR <sup>↑</sup>	SSIM <sup>↑</sup>	FLIP <sup>↓</sup>	VRAM <sup>↓</sup>
gsplat	–	–	–	7.3GB
gsplat <sup>†</sup>	32.87	0.969	0.034	4.7GB
V3DG	52.30	0.998	0.004	10.3GB
Ours w/o MLP	45.21	0.995	0.027	4.2GB
Ours	40.95	0.995	0.029	3.1GB

## References

- [1] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. FLIP: A difference evaluator for alternating images. *PACMCGIT*, 3(2), 2020. 6, 11
- [2] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ fps. In *NeurIPS*, 2024. 2
- [3] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. EA-GLES: Efficient accelerated 3D Gaussians with lightweight encodings. In *ECCV*, pages 54–71, 2024. 2, 3, 4
- [4] Álvaro González. Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices. *Math. Geosci.*, 42: 49–64, 2010. 4
- [5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 1, 2, 4, 6
- [6] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. *ACM TOG*, 43(4), 2024. 1, 3
- [7] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov chain Monte Carlo. In *NeurIPS*, 2024. 2
- [8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6
- [9] Jonas Kulhanek, Marie-Julie Rakotosaona, Fabian Manhardt, Christina Tsalicoglou, Michael Niemeyer, Torsten Sattler, Songyou Peng, and Federico Tombari. LODGE: Level-of-detail large-scale Gaussian splatting with efficient rendering. In *NeurIPS*, 2025. 1, 3
- [10] Jiwon Lee, Yunjae Lee, Youngeun Kwon, and Minsoo Rhu. Characterization and analysis of the 3D Gaussian splatting rendering pipeline. *IEEE Comput. Archit. Lett.*, 24(1):13–16, 2025. 5
- [11] Shiyong Liu, Xiao Tang, Zhihao Li, Yingfan He, Chongjie Ye, Jianzhuang Liu, Binxiao Huang, Shunbo Zhou, and Xiaofei Wu. OccluGaussian: Occlusion-aware Gaussian splatting for large scene reconstruction and rendering. In *ICCV*, pages 26643–26652, 2025. 3, 4
- [12] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. In *CVPR*, pages 20654–20664, 2024. 3
- [13] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D scene representation via self-organizing Gaussian grids. In *ECCV*, pages 18–34, 2025. 2
- [14] Thomas Müller. tiny-cuda-nn, 2021. 4
- [15] K L Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. CompGS: Smaller and faster Gaussian splatting with vector quantization. In *ECCV*, pages 330–349, 2024. 2
- [16] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. RadSplat: Radiance field-informed Gaussian splatting for robust real-time rendering with 900+ fps. In *3DV*, pages 134–144, 2025. 3, 4
- [17] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3D Gaussian splatting. *PACMCGIT*, 7(1), 2024. 2
- [18] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM TOG*, 43(4), 2024. 2
- [19] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. *TPAMI*, 2025. 1, 3
- [20] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kotschieder. Revising densification in Gaussian splatting. In *ECCV*, pages 347–362, 2024. 2
- [21] Markus Schütz, Christoph Peters, Florian Hahlbohm, Elmar Eisemann, Marcus Magnor, and Michael Wimmer. Splatshop: Efficiently editing large Gaussian splat models. *Comput. Graph. Forum*, 2025. 3
- [22] Yunji Seo, Young Sun Choi, HyunSeung Son, and Youngjung Uh. FLoD: Integrating flexible level of detail into 3D Gaussian splatting for customizable rendering. *ACM TOG*, 44(4), 2025. 1, 3

- [23] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. 4, 12
- [24] Jonathan Tremblay, Moustafa Meshry, Alex Evans, Jan Kautz, Alexander Keller, Sameh Khamis, Thomas Müller, Charles Loop, Nathan Morrical, Koki Nagano, Towaki Takikawa, and Stan Birchfield. RTMV: A ray-traced multi-view synthetic dataset for novel view synthesis. In *ECCVW*, 2022. 6, 7, 11, 15
- [25] Karan Vagadia. Donut 3d model. <https://www.cgtrader.com/free-3d-models/food/beverage/realistic-donut-fa8648b6-f2f6-4fb3-93d9-ce531d2ba013>, 2021. 3D model, accessed on May 20, 2024. 6, 7, 12, 15
- [26] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6
- [27] Zhangyang Xiong, Chenghong Li, Kenkun Liu, Hongjie Liao, Jianqiao Hu, Junyi Zhu, Shuliang Ning, Lingteng Qiu, Chongjie Wang, Shijie Wang, Shuguang Cui, and Xiaoguang Han. MVHumanNet: A large-scale dataset of multi-view daily dressing human captures. In *CVPR*, pages 19801–19811, 2024. 6, 7, 11, 15
- [28] Xijie Yang, Linning Xu, Lihan Jiang, Dahua Lin, and Bo Dai. Virtualized 3D Gaussians: Flexible cluster-based level-of-detail system for real-time rendering of composed scenes. In *SIGGRAPH*, 2025. 1, 3, 5, 6, 7, 11, 12, 17
- [29] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *JMLR*, 26(34):1–17, 2025. 1, 3, 6
- [30] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian splatting. In *CVPR*, pages 19447–19456, 2024. 3
- [31] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. LP-3DGS: Learning to prune 3D Gaussian splatting. In *NeurIPS*, 2024. 2
- [32] Brent Zoomers, Maarten Wijnants, Ivan Molenaers, Joni Vanherck, Jeroen Put, Lode Jorissen, and Nick Michiels. PRoGS: Progressive rendering of Gaussian splats. In *WACV*, pages 3118–3127, 2025. 3, 4



Asset	GT			Ours			$\Delta$ Passed (%)
	Passed	Used	Used (%)	Passed	Used	Used (%)	
100831	313,257	151,748	48.4%	199,207	148,328	74.5%	-36.41%
100833	273,513	134,121	49.0%	183,906	130,578	71.0%	-32.76%
100835	302,469	153,949	50.9%	206,470	151,189	73.2%	-31.74%
100837	331,681	164,372	49.6%	214,328	159,772	74.5%	-35.38%
100839	295,192	146,948	49.8%	202,081	143,489	71.0%	-31.54%
100841	301,718	154,509	51.2%	194,411	151,447	77.9%	-35.57%
100843	278,513	141,775	50.9%	198,186	138,574	69.9%	-28.84%
100845	824,204	393,820	47.8%	494,592	387,509	78.3%	-39.99%
100847	334,039	158,388	47.4%	207,953	154,531	74.3%	-37.75%
100849	382,225	182,777	47.8%	235,006	177,868	75.7%	-38.52%
100851	318,754	149,637	46.9%	201,116	145,765	72.5%	-36.91%
100853	368,617	180,419	48.9%	241,633	176,145	72.9%	-34.45%
100855	286,111	140,272	49.0%	189,134	137,433	72.7%	-33.90%
100857	281,235	136,311	48.5%	181,743	132,082	72.7%	-35.38%
100859	229,308	109,789	47.9%	148,576	107,207	72.2%	-35.21%
100861	264,017	128,745	48.8%	170,439	125,885	73.9%	-35.44%
bonsai	319,339	209,366	65.6%	284,781	204,559	71.8%	-10.82%
crispy	264,978	191,948	72.4%	237,725	191,353	80.5%	-10.29%
desk	425,546	244,866	57.5%	339,417	237,974	70.1%	-20.24%
dinosaur	221,706	144,505	65.2%	207,475	143,451	69.1%	-6.42%
fall	276,574	147,584	53.4%	202,262	144,641	71.5%	-26.87%
house	273,216	168,544	61.7%	240,408	167,239	69.6%	-12.01%
oak	257,670	205,816	79.9%	246,392	205,118	83.2%	-4.38%
pine	264,686	156,355	59.1%	202,004	152,392	75.4%	-23.68%
donut	44,106	26,267	59.6%	29,964	26,052	86.9%	-32.06%

Table 5. Per-asset statistics. We display the number of Gaussians passed to the rasterizer, as well as the Gaussians that actually contribute to the final rendering, both in absolute numbers and as percentages relative to the total number of Gaussians passed. We then show the difference in Gaussians passed between the ground truth and our method in the last column. We divide the assets into their three respective datasets. The first 16 entries are from the MVHumanNet dataset [27], the next eight from the RTMV [24] dataset, and the donut from [25]. We prune a significant number of Gaussians across all assets. For oak and dinosaur, the lower gains can be dedicated to the higher-frequency visibility functions, which comprise the majority of the asset. While this could be improved by using frequency encodings, we argue that the slowdown in general for these specific assets is not worth the slight increase for a small subset of assets.

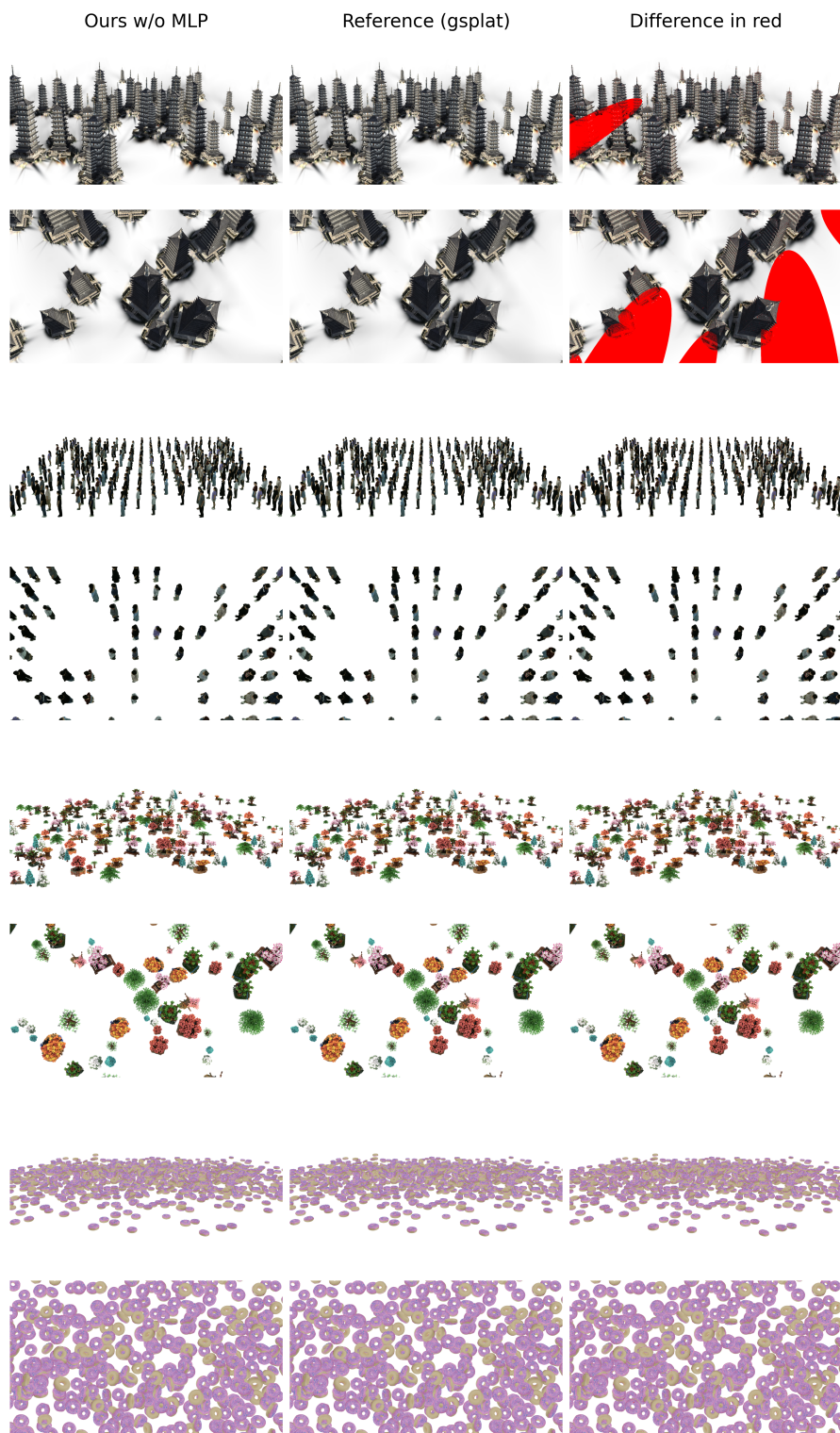


Figure 13. Left: ours w/o MLP; center: reference rendering from gsplat; right: difference image where red pixels indicate any difference between left and center. For the TEMPLES layout, there are large Gaussians that cover large parts of the scene, colored in red. While visually there is little to no difference, removing these Gaussians results in a small hue shift that significantly affects image metrics, especially PSNR.

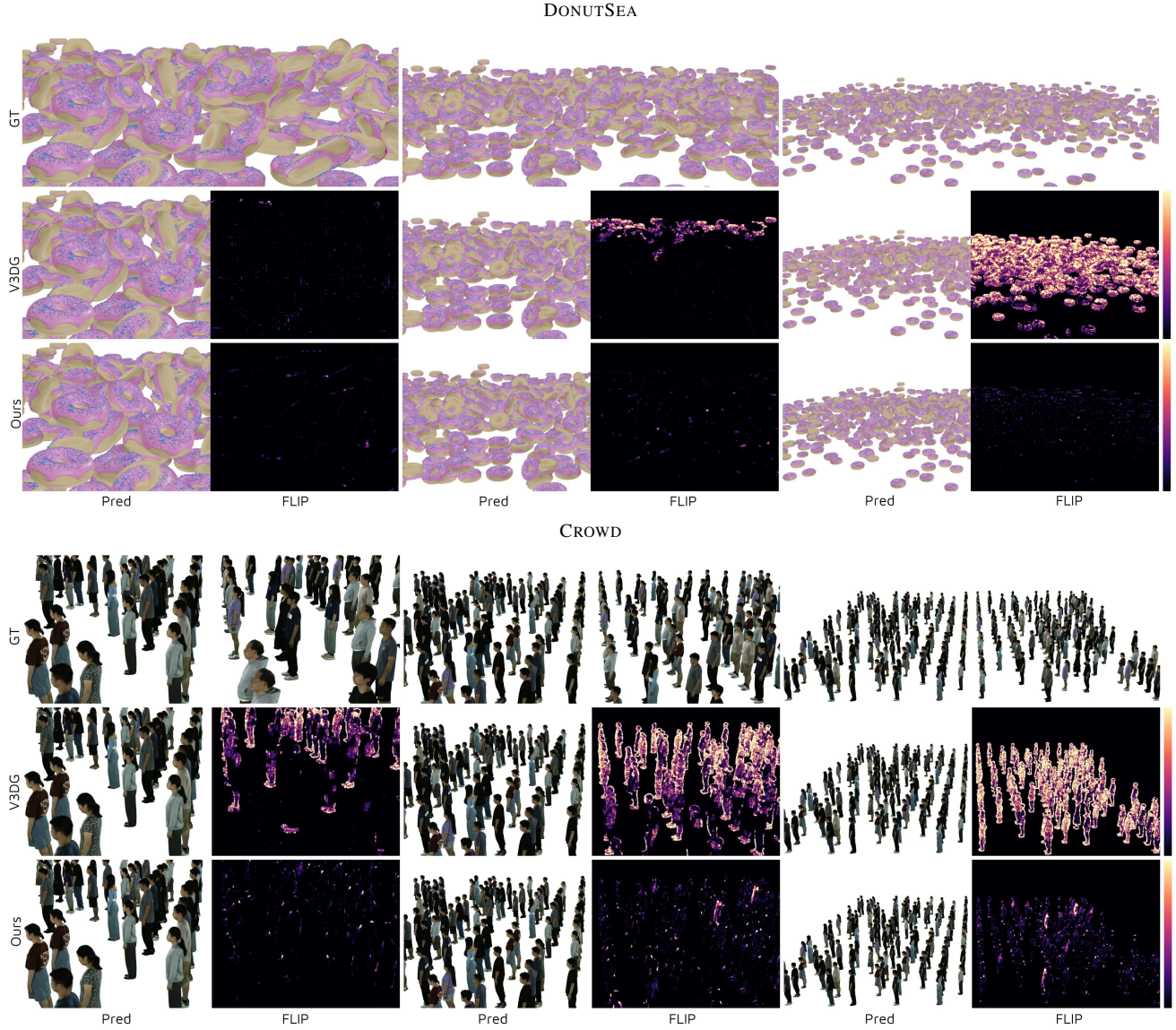


Figure 14. Shows GT render using gsplat, V3DG, and our method for close, medium, and far distances on the DONUTSEA and CROWD layouts. For V3DG and ours, we show the prediction on the left and FLIP on the right. We scaled FLIP by a factor of five to facilitate better visual comparison. These scenes have been downsampled by a factor of two compared to the original layouts provided by V3DG [28] to ensure that all methods can be tested without exceeding VRAM limits.



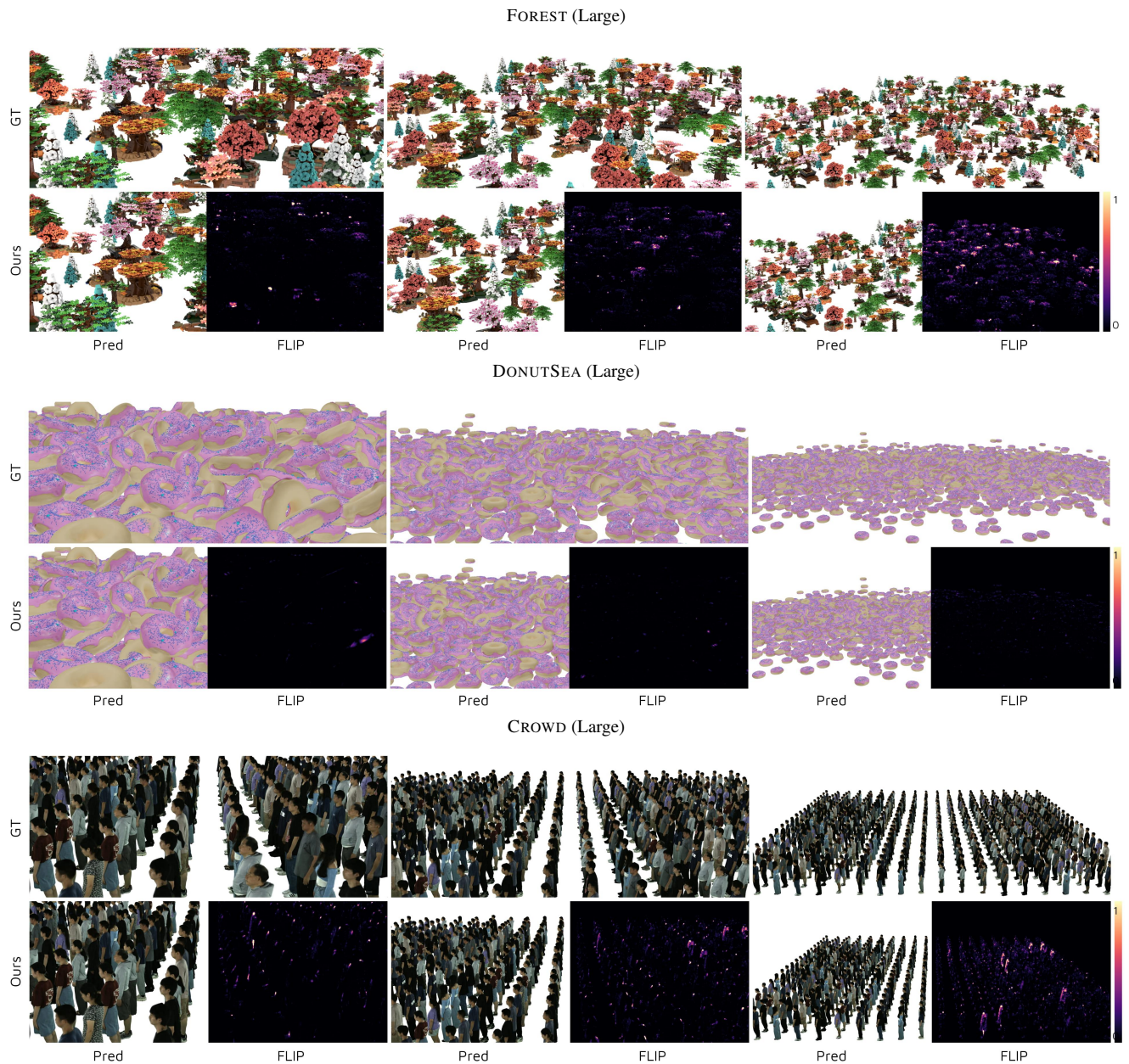


Figure 15. Shows GT render using our method with and without MLP(solely using instanced rasterizer) for close, medium, and far distances on the FOREST, DONUTSEA, and CROWD layouts. For ours, we show the prediction on the left and FLIP on the right. These renderings show the original layout files without downscaling (individual assets remain centered after removing all Gaussians with an opacity lower than  $\frac{1}{255}$ ). We scaled FLIP by a factor of five to facilitate visual comparison.