
IBGS: Image-Based Gaussian Splatting

—Supplementary Material—

Hoang Chuong Nguyen¹ Wei Mao Jose M. Alvarez² Miaomiao Liu¹

¹Australian National University ²NVIDIA

hoangchuong.nguyen@anu.edu.au miaomiao.liu@anu.edu.au

wei.mao.research@gmail.com josea@nvidia.com

A More Implementation Details

Mixed CUDA and PyTorch Implementation. We implement our residual prediction in PyTorch and modify the CUDA kernels of 3DGS [9] to support finding the ray-Gaussian intersection, projecting the intersection to the source view, performing visibility check and warping colors from source views. This custom CUDA kernels enables different processing for different pixels. For example, we can extract features from different source views based on the pixel’s visibility in each view.

Choosing the Median Gaussians. To find the K median intersections, we maintain two buffers of size $\frac{K}{2}$. The first buffer stores the $\frac{K}{2}$ intersections encountered *before* the accumulated weight w reaches 0.5, using a FIFO (first-in, first-out) strategy. In particular, while $w \geq 0.5$, we update this buffer by appending the new intersection if it is not full, or by replacing the oldest intersection with the new one otherwise. Once $w < 0.5$, we simply append the next $\frac{K}{2}$ intersections to the second buffer. The union of these two buffers then yields the K intersections whose accumulated weight straddles 0.5.

Obtaining Depth Maps of Source Views. We leverage the precomputed depth maps of each source view used in the visibility check (Eq. 15 in the main paper). Specifically, during training, after a training iteration for a given view completes, we store that view’s rendered depth map and later use it for visibility checking. This avoids recomputing depth maps for the source views in every training iteration, which is time-consuming. At test time, depth maps of source images are rendered on-the-fly.

B Additional Results

B1 Joint Novel-View Synthesis and 3D Reconstruction Evaluation.

In Tab. 1, we compare the performance of different methods in both novel-view synthesis (NVS) and 3D reconstruction on six scenes from the Tanks&Temples dataset. For this experiment, we apply TSDF [4] as a mesh extraction approach to all methods. Specifically, we first render depth maps, then extract meshes using TSDF fusion, and finally compute the F1 score for the extracted meshes. The results show that PGSR [3] achieves the best 3D reconstruction but lags behind our method in NVS performance by **2.68** dB on average. This is because PGSR applies additional geometric regularization (such as enforcing planar Gaussians and forward-backward projection consistency), which enhances reconstructed geometry but compromises image quality [6, 7]. On the other hand, scene geometry and image quality are tightly coupled in our method: higher geometric accuracy leads to more accurate appearance features extracted from the source views. As a result, our method strikes a good balance between the two tasks, and on average achieves the best image quality and the second-best 3D reconstruction performance.

Table 1: Comparison of different methods in both novel-view synthesis and 3D reconstruction. We report the F1 scores (with TSDF fusion) for 3DGS and 2DGS from [7], and those for GOF and RaDe-GS are from [23].

Scenes	3DGS [9]		2DGS [7]		GOF [22]		RaDe-GS [23]		PGSR [3]		Ours	
	PSNR	F1 score	PSNR	F1 score	PSNR	F1 score	PSNR	F1 score	PSNR	F1 score	PSNR	F1 score
Barn	27.99	0.13	<u>28.79</u>	0.41	26.56	0.37	26.46	0.43	28.00	0.66	30.19	<u>0.56</u>
Caterpillar	<u>24.82</u>	0.08	24.23	0.23	21.86	0.21	22.71	0.26	22.44	0.44	26.06	<u>0.31</u>
Courthouse	23.33	0.09	<u>23.51</u>	<u>0.16</u>	20.24	0.11	19.72	0.11	20.58	0.20	24.46	0.06
Ignatius	<u>23.95</u>	0.04	23.82	0.51	20.90	0.63	21.30	<u>0.73</u>	21.63	0.81	24.02	0.63
Meetingroom	<u>26.89</u>	0.01	26.15	0.17	24.45	<u>0.23</u>	25.16	0.17	23.81	0.33	27.18	0.22
Truck	25.01	0.19	<u>26.85</u>	0.45	23.11	0.50	23.61	0.53	24.85	0.66	25.46	<u>0.55</u>
average	25.33	0.09	<u>25.56</u>	0.32	22.85	0.34	23.16	0.37	23.55	0.52	26.23	<u>0.39</u>

Table 2: Comparison with GStex [15] and Texture-GS [19].

Dataset	DTU				Mip-NeRF 360		
Method Metric	PSNR	SSIM	LPIPS	Memory	PSNR	SSIM	LPIPS
3DGS [9]	32.87	<u>0.956</u>	0.052	80MB	<u>28.96</u>	<u>0.869</u>	0.185
2DGS [7]	32.22	0.940	0.090	41MB	28.30	0.858	0.162
Texture-GS [19]	30.53	0.920	0.083	88MB	–	–	–
GStex [15]	<u>32.87</u>	<u>0.956</u>	0.038	53MB	28.81	0.867	0.142
Ours	33.26	0.961	<u>0.051</u>	17MB	30.04	0.886	<u>0.158</u>

B2 Comparison with GStex [15] and Texture-GS [19]

Tab. 2 shows the comparison between our method, Texture-GS[19], and GStex [15] on object-level scenes in the DTU [8] dataset and GStex’s 7 selected scenes in the Mip-NeRF 360 [1] dataset. In both datasets, our method achieves the best PSNR and SSIM compared to Texture-GS [19], which leverages global texture maps, and GStex [15], which learns per-Gaussian texture maps and requires more storage memory than our method.

B3 Comparison with Image-Based Neural Rendering Method [16]

Tab. 3 presents the NVS performance, rendering speed (in FPS), storage memory of our method and an image-based neural rendering method [16] on two scenes in the Shiny [17] dataset. Note that the FPS of both methods is measured using a single RTX 4090, and the reported memory already includes the memory for storing the source images. Our method achieves comparable performance to [16] while being significantly faster in terms of the rendering FPS. Unlike [16] which requires densely sampling 3D points along a camera ray and projecting all of them to the source views, our method projects only a sparse set of K median intersections (i.e., $K = 4$) to the source views, thereby enabling fast rendering speed. Additionally, [16] leverages a large transformer model to predict full pixel colors, which further adds overhead to the rendering time. In contrast, our method predicts color residuals instead of full colors, which allows us to utilize a lightweight network for this task. Moreover, our Gaussians consumes less storage compared to the network used in [16].

B4 Comparison with Additional Gaussian Splatting Methods

In Tab. 4, we compare our method with additional approaches that focus on improving different aspects of Gaussian Splatting, such as Gaussian densification strategies [21], 3D reconstruction [3, 7],

Table 3: Comparison with an image-based neural rendering method [16] in the Shiny dataset

Scene	CD					Lab				
	PSNR	SSIM	LPIPS	FPS	Memory	PSNR	SSIM	LPIPS	FPS	Memory
Light field neural rendering [10]	35.25	0.989	0.041	0.0025	104	35.28	0.989	0.066	0.0025	97
Ours	35.21	0.955	0.062	56	69	35.07	0.967	0.056	58	66

Table 4: Comparison between our method and previous works in three datasets. We report two released results from TexturedGaussian [2], with and without total memory usage.

Dataset Method Metric	Mip-NeRF 360				Tanks&Temples (TNT)				Deep blending			
	PSNR	SSIM	LPIPS	Mem	PSNR	SSIM	LPIPS	Mem	PSNR	SSIM	LPIPS	Mem
Mip-NeRF 360 [1]	27.69	0.792	0.237	8.6	22.22	0.759	0.257	8.6	29.40	0.901	0.245	8.6
Instant-NGP [12]	25.30	0.671	0.371	13	21.72	0.723	0.330	13	23.62	0.797	0.423	13
3DGS [9]	27.69	0.825	0.203	764	23.11	0.840	0.184	415	29.53	0.904	0.242	745
SuperGauss [18]	27.31	0.815	0.209	1021	23.72	0.847	0.179	502	28.83	0.901	0.250	762
TexturedGauss [2]	27.35	0.827	0.186	–	24.26	0.854	0.168	–	28.33	0.891	0.270	–
TexturedGauss* [2]	27.26	–	–	1047	24.28	–	–	691	28.52	–	–	668
StopThePop [13]	27.28	0.814	0.213	717	23.21	0.843	0.173	418	29.86	0.904	0.234	637
AbsGS [21]	27.49	0.820	0.191	728	23.73	0.853	0.162	304	29.67	0.902	0.236	444
PGSR [3]	27.25	0.833	<u>0.178</u>	880	22.96	0.857	0.146	351	28.55	0.882	0.257	790
2DGS [7]	27.03	0.804	0.239	484	23.13	0.833	0.211	205	29.49	0.903	0.256	357
Scaffold-GS [11]	27.66	0.807	0.236	203	23.96	0.853	0.177	87	30.21	0.906	0.254	66
Octree-GS (3DGS) [14]	27.65	0.815	0.220	419	24.17	0.858	0.161	384	29.65	0.901	0.257	180
Spec-Gaussian [20]	28.18	0.835	0.176	886	23.86	0.854	0.166	386	29.31	0.903	0.244	586
Eagles [5]	27.23	0.810	0.240	54	23.37	0.840	0.200	29	29.86	0.910	0.250	52
Ours	<u>28.33</u>	<u>0.837</u>	0.186	291	<u>24.84</u>	<u>0.869</u>	<u>0.148</u>	143	<u>30.12</u>	0.912	<u>0.237</u>	198
Ours + [21]	28.48	0.841	0.180	317	24.92	0.872	0.146	150	30.07	<u>0.911</u>	0.234	209

Table 5: Comparison with Spec-Gauss [20] on Shiny dataset.

Scene Method Metric	Guitars (specular highlight)			Lab (reflection)			CD (diffraction)		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Spec-Gauss [20]	30.62	0.955	0.120	30.53	0.946	0.103	30.69	0.954	0.081
Ours	35.78	0.954	0.104	35.07	0.967	0.056	35.21	0.955	0.062

and Gaussians compression [5]. Overall, our method can render images with higher quality compared to the previous works as we consistently achieve the best SSIM across all datasets. Moreover, we show that integrating the Gaussian densification strategy of [21] into our pipeline further improves the performance. This is an example demonstrating that we can further boost the performance of our method by incorporating techniques proposed in the orthogonal works into our proposed pipeline.

We also compare IBGS with Spec-Gaussian [20] on the Shiny dataset in Tab. 5. IBGS achieves significantly better performance in almost all metrics, outperforming Spec-Gaussians by at least **4.52** dB. Unlike Spec-Gaussians, which focuses solely on modeling the specular component of scene appearance, IBGS is not limited to modeling specular color and can also capture various types of view-dependent effects, such as reflection and diffraction.

B5 More Qualitative Results

We show additional qualitative comparisons between our method, 3DGS [9], SuperGauss [18], GStex [15] and Texture-GS [19] in Figs. 1, 2, and 3. Our method can render images with both high-frequency details and view-dependent color, which is challenging for the other methods. We also provide visualizations of the rendered depths maps and normal maps in Fig. 4 and Fig 5.

C Additional Ablation Studies

C1 Input/Output of the Residual Prediction Network

We conduct an ablation study on the input and output of the residual prediction network and show the results in Tab. 6. Here we have four variants of our method.

Input: base color, color differences. Output: color residual. This is our original design that uses the base color and the color differences as input to predict the color residual, which is later added to the base color. This design choice yields the best performance compared to the others.

Table 6: Ablation study on the colors input/output of the residual prediction network. Note that although the view direction is not listed in the table, we always have it as an input into the model.

Input/output of the residual prediction network		Mip-NeRF 360			Tanks&Temples			Deep Blending		
Input	Output	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Base color \mathbf{c} , color differences $\{\Delta\mathbf{c}_m\}_{m=1}^M$	Residual $\Delta\mathbf{C}$	28.33	0.837	0.186	24.84	0.869	0.148	30.12	0.912	0.237
Base color \mathbf{c} , source colors $\{\mathbf{c}_m\}_{m=1}^M$	Residual $\Delta\mathbf{C}$	28.21	0.837	<u>0.187</u>	24.61	<u>0.867</u>	<u>0.150</u>	29.99	0.910	0.242
Base color \mathbf{c} , source colors $\{\mathbf{c}_m\}_{m=1}^M$	Final image $\mathbf{C}^{\text{final}}$	28.08	0.835	0.193	24.58	0.866	0.159	30.03	0.912	0.240
Color differences $\{\Delta\mathbf{c}_m\}_{m=1}^M$	Residual $\Delta\mathbf{C}$	28.33	<u>0.834</u>	0.191	<u>24.82</u>	0.866	0.154	30.12	<u>0.911</u>	0.237

Table 7: Ablation study on the SH degree.

Method	SH degree	Mip-NeRF 360					Tanks and Temples					Deep blending				
		PSNR	SSIM	LPIPS	#Gauss	Mem	PSNR	SSIM	LPIPS	#Gauss	Mem	PSNR	SSIM	LPIPS	#Gauss	Mem
3DGS [9]	0	27.08	0.815	0.214	3.2	208	22.73	0.832	0.196	1.8	114	29.47	0.904	0.242	3.1	199
	1	27.30	0.821	0.208	3.2	318	22.79	0.833	0.192	1.7	173	29.71	0.905	0.240	3.1	308
	2	27.55	0.824	0.204	3.2	505	23.06	0.838	0.187	1.7	272	29.64	0.905	0.240	3.2	496
	3	27.69	0.825	0.203	3.2	764	23.11	0.840	0.184	1.8	415	29.53	0.904	0.242	3.1	745
Ours	0	28.22	0.833	0.192	1.5	143	24.66	0.865	0.154	0.7	73	30.06	0.911	0.238	1.1	93
	1	28.28	0.835	0.188	1.6	197	24.79	0.867	0.150	0.7	98	30.15	0.912	0.236	1.1	131
	2	28.41	0.837	0.185	1.6	292	24.82	0.869	0.149	0.8	143	30.03	0.912	0.236	1.1	198
	3	28.35	0.835	0.187	1.6	417	24.90	0.870	0.146	0.8	205	29.84	0.911	0.239	1.1	284

Input: base color, source colors. Output: color residual. Here, we modify our method by using the source colors as input in place of the color differences. As a result, the image quality drops compared to our original design.

Input: base color, source colors. Output: final image. This variant has similar inputs to the previous one but changes the final output to the full color. This design results in worse image quality compared to predicting color residuals (i.e., the second row in Tab. 6). These results verify the effectiveness of our design: using the color difference as input and predicting the color residual leads to better performance than directly leveraging the full color as the network’s input/output.

Input: color differences. Output: color residual. For the last variant, we remove the base color from the network’s input. Consequently, we observe performance degradation in the Mip-NeRF 360 and Tanks&Temples datasets. This is because using the full base image as input provides the network with contextual information that is missing from the input color differences. Thus, including this information as input yields better performance.

C2 Resolution of the Predicted Color Residual Map $\Delta\mathbf{C}$

In this ablation study, we train our method with varying resolutions of the predicted color residual map and demonstrate that our method can achieve faster rendering with a trade-off in image quality. Specifically, after extracting features from the source views, we downsample the original extracted feature map by a factor $s \in \{1.0, 0.5, 0.25\}$ and use them as input of the residual prediction network to predict a residual map at the same downsampled size. After that, we upsample the predicted residual map to the original image resolution and add it to the rendered base image. The result in Tab. 8 reveals that as the resolution decreases, our method speeds up rendering time while still significantly outperforming 3DGS [9] and SuperGauss [18] in terms of image quality.

Table 8: Ablation study on the resolution of the predicted color residual map.

Method	Residual downsample factor	Mip-NeRF360					Tanks&Temples					Deep blending				
		PSNR	SSIM	LPIPS	FPS	Mem	PSNR	SSIM	LPIPS	FPS	Mem	PSNR	SSIM	LPIPS	FPS	Mem
3DGS [9]	×	27.69	0.825	0.203	196	764	23.11	0.840	0.184	248	415	29.53	0.904	0.242	186	745
SuperGauss [18]	×	27.31	0.815	0.209	51	1021	23.72	0.847	0.179	86	502	28.83	0.901	0.250	49	762
Ours	1	28.33	0.837	0.186	25	291	24.84	0.869	0.148	48	143	30.12	0.912	0.237	35	197
Ours	0.5	<u>28.24</u>	<u>0.832</u>	<u>0.195</u>	38	284	<u>24.74</u>	<u>0.861</u>	<u>0.162</u>	58	143	<u>30.11</u>	<u>0.911</u>	<u>0.240</u>	49	189
Ours	0.25	28.15	0.824	0.211	40	289	24.61	0.849	0.188	62	142	30.09	0.911	0.242	53	192

Table 9: Ablation study on the exposure correction module.

Method	Tanks and Temples			Mip-NeRF360		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
With exposure correction	24.84	0.869	0.148	28.31	0.835	0.188
Without exposure correction	24.28	0.866	0.152	28.33	0.837	0.186

Table 10: Training time comparison (minutes). All models are trained using a single RTX4090.

Method\Dataset	MipNeRF-360	Tanks & Temples (TNT)	Deep blending
3DGS [9]	19	11	19
SuperGaussian [18]	55	30	60
Ours	44	21	39

C3 Ablation Study on the SH degree

In this section, we compare the sensitivity of our method and 3DGS [9] to the SH degree and show that our method can use a low SH degree to further reduce the storage memory while incurring a smaller performance drop compared to 3DGS. Tab. 7 shows that, when reducing the SH degree from 3 to 0, the PSNR drop of our method is **79%** and **37%** smaller than that of 3DGS in the Mip-NeRF 360 and the TNT datasets, respectively. This demonstrates that our method is less sensitive to the SH degree compared to 3DGS, and thus can use a smaller degree to further save storage memory while still achieving significantly better image quality.

C4 Ablation Study on the exposure correction module

Tab. 9 shows an ablation study for the exposure correction module. As discussed in [1], the camera auto-exposure setting was enabled while capturing images in the Tanks and Temples datasets. Thus, applying our proposed exposure correction method to this dataset yields a significant PSNR gain of **0.56 db**. In practice, the exposure correction module can be activated by default without concern for performance degradation, as it has minimal impact when there is no exposure variation across the training views. This is demonstrated via the results on the MipNeRF-360 dataset in Tab. 9, showing that the image quality remains largely unchanged when the exposure correction is applied.

D Limitations

While improving the image quality significantly, IBGS achieves lower rendering speed (see Tab. 8) and requires longer training time compared to the vanilla 3DGS [9] (see Tab. 10) due to the additional computations for predicting the color residuals. Moreover, IBGS could encounter memory bottlenecks when handling large scenes as it may require pre-loading a substantial number of source images into GPU VRAM to avoid the overhead associated with on-the-fly memory allocation for these images. This approach leads to higher VRAM usage of IBGS compared to that of 3DGS and SuperGaussian, as illustrated in Tab. 11. A feasible solution to mitigate this limitation is to load each image into VRAM only when it is selected as a source view needed for rendering. Nevertheless, this approach introduces a trade-off between speed and memory usage, as dynamic VRAM allocation can adversely affect the rendering speed of IBGS.

Table 11: Comparison in terms of peak GPU VRAM consumption during inference (GB).

Method\Dataset	MipNeRF-360	Tanks & Temples (TNT)	Deep blending
3DGS	2.03	1.25	2.13
SuperGaussians	2.76	1.45	2.44
Ours	6.12	2.97	5.36

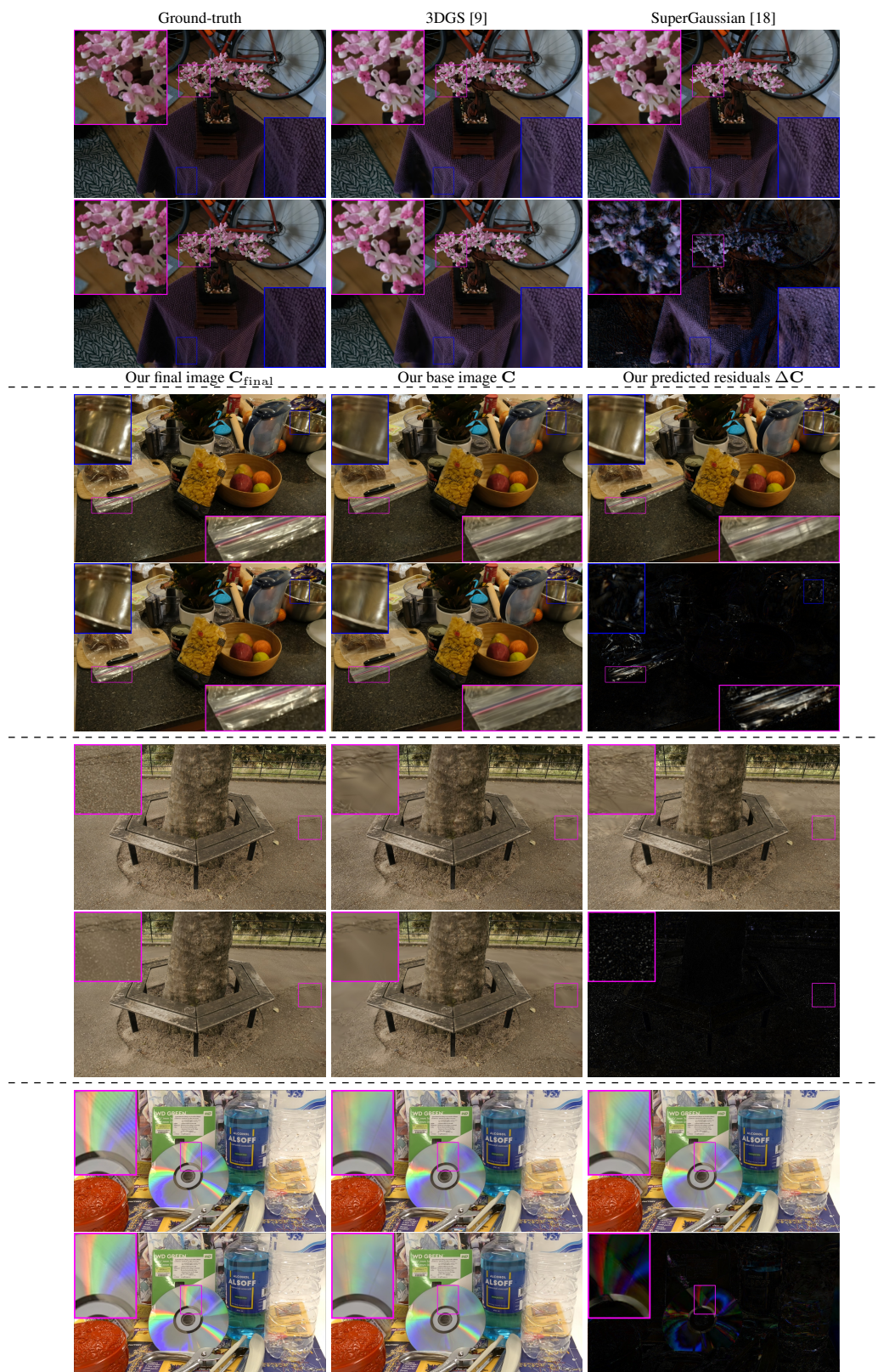


Figure 1: Qualitative comparison with 3DGS [9] and SuperGauss [18].

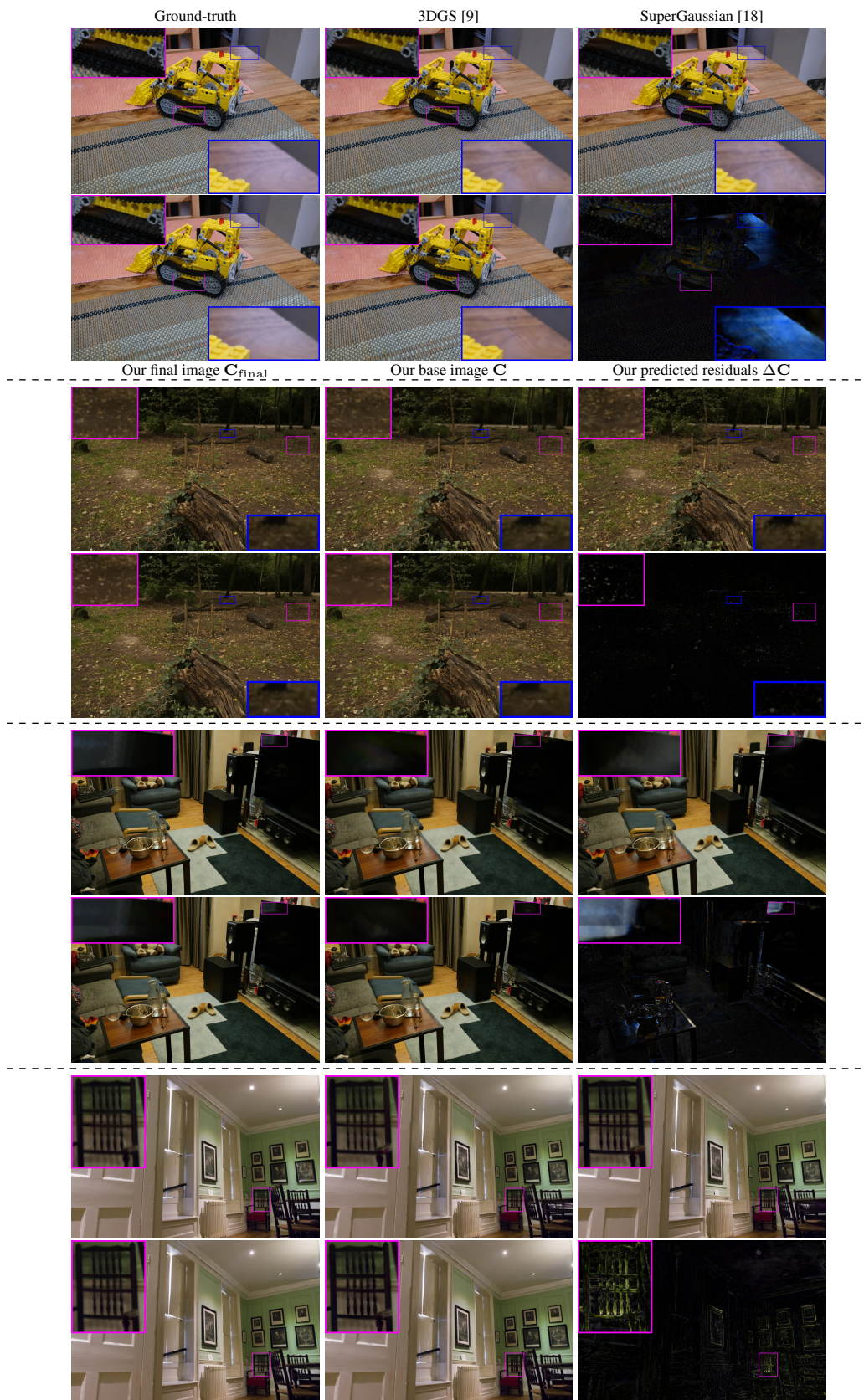


Figure 2: Qualitative comparison with 3DGS [9] and SuperGauss [18].

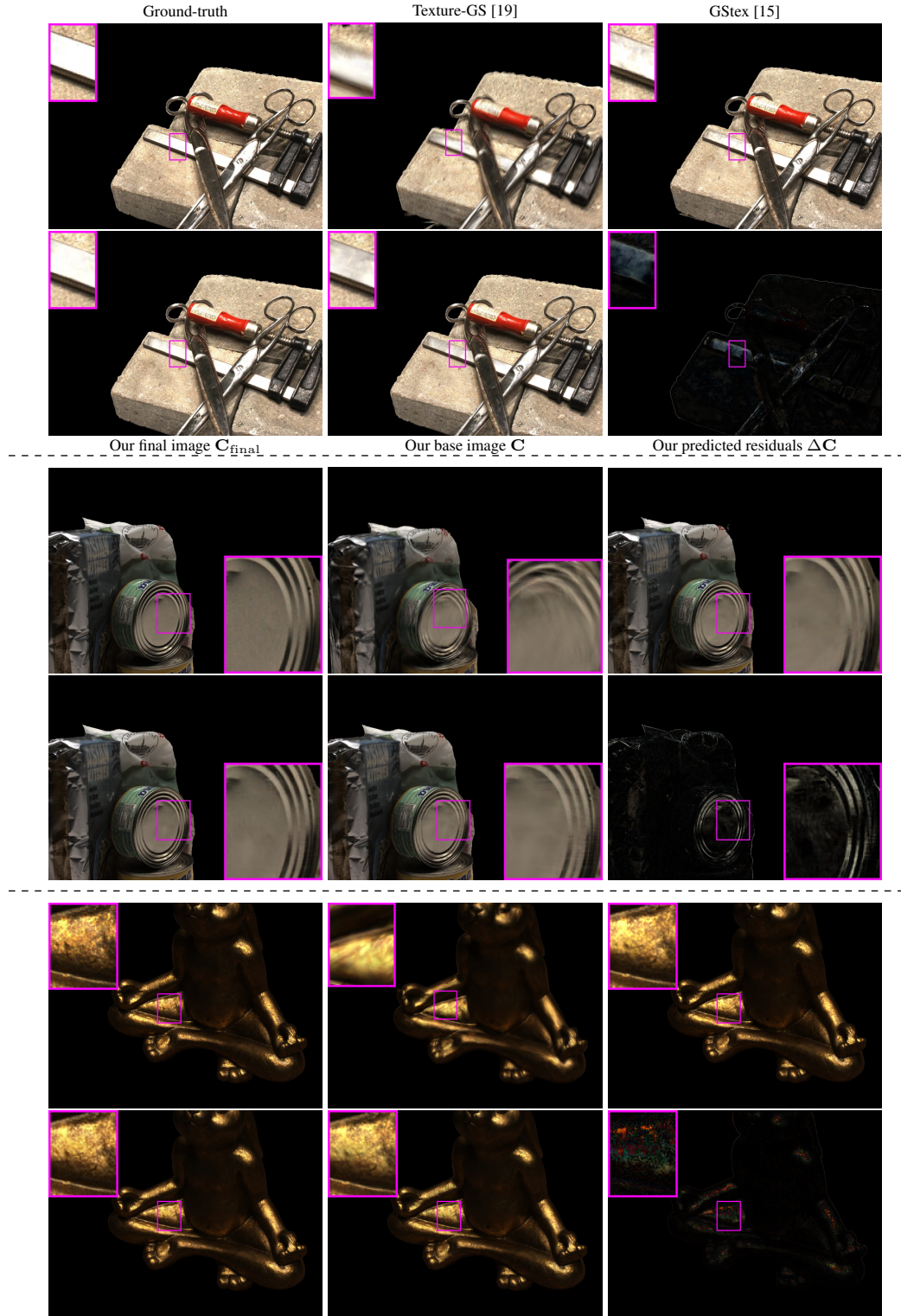


Figure 3: Qualitative comparison with Texture-GS [19] and GStex [15] on the DTU dataset.

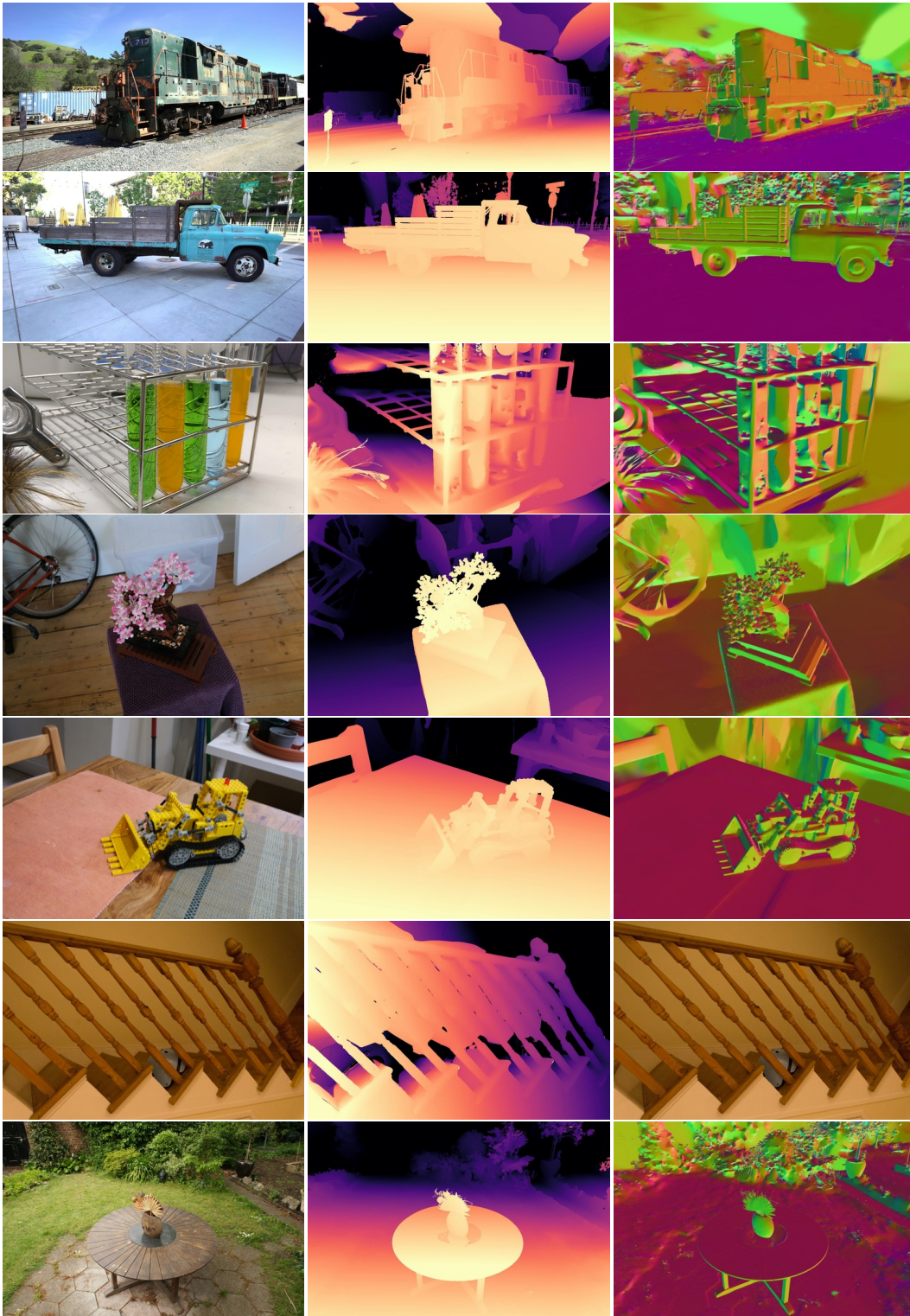


Figure 4: Ground-truth images (left), rendered depths (middle) and rendered normals (right).

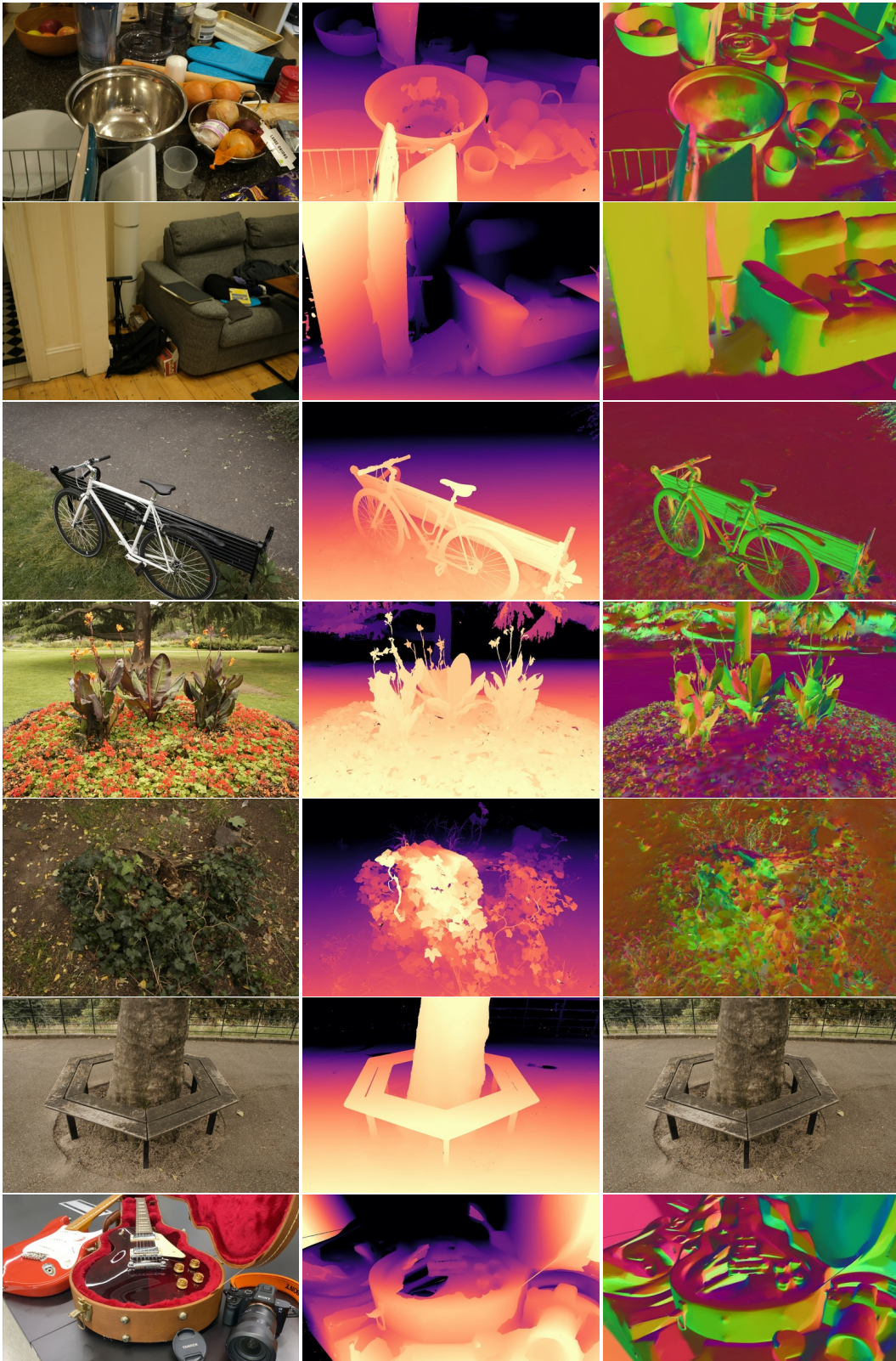


Figure 5: Ground-truth images (left), rendered depths (middle) and rendered normals (right).

References

- [1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.
- [2] Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, et al. Textured gaussians for enhanced 3d scene appearance modeling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [3] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
- [4] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.
- [5] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*, pages 54–71. Springer, 2024.
- [6] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.
- [7] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.
- [8] 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] 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.
- [10] Marc Levoy and Pat Hanrahan. Light field rendering. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 441–452. 2023.
- [11] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.
- [12] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.
- [13] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024.
- [14] 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. *arXiv preprint arXiv:2403.17898*, 2024.
- [15] Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N Kutulakos, and David B Lindell. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3508–3518. IEEE, 2025.
- [16] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8269–8279, 2022.
- [17] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021.
- [18] Rui Xu, Wenyue Chen, Jiepeng Wang, Yuan Liu, Peng Wang, Lin Gao, Shiqing Xin, Taku Komura, Xin Li, and Wenping Wang. Supergaussians: Enhancing gaussian splatting using primitives with spatially varying colors. *arXiv preprint arXiv:2411.18966*, 2024.

- [19] Tian-Xing Xu, Wenbo Hu, Yu-Kun Lai, Ying Shan, and Song-Hai Zhang. Texture-gs: Disentangling the geometry and texture for 3d gaussian splatting editing. In *European Conference on Computer Vision*, pages 37–53. Springer, 2024.
- [20] Ziyi Yang, Xinyu Gao, Yang-Tian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37:61192–61216, 2024.
- [21] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1053–1061, 2024.
- [22] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.
- [23] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024.