Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering

Supplementary Material

1. Video Comparison

To better demonstrate the improvement of our algorithm in quality and speed for different resolutions, we include a video comparing our results with the original 3D Gaussian Splatting[3] at multiple scenes from different views and resolutions.

2. Details of Gaussian Aggregation Algorithm

Due to the space constraint of the main paper, some details of the Gaussian aggregation process are omitted. In this section, we will elaborate further with some examples to help the readers understand and reproduce our work. The process consists of the following steps:

Render at Lower Resolution. Since we want to insert large Gaussians that are of appropriate size to be rendered at lower resolutions, we need to aggregate small Gaussians to form large Gaussians. Pixel coverage is used to determine whether a Gaussian is too small, we need to render all Gaussians first to calculate their pixel coverage at all training cameras. For all coarse levels $l_m = [2, l_{max}]$, we render all Gaussians from $[1, l_m - 1]$ at $4^{l_m - 1}$ times downsampled resolution. For example, we render all Gaussians from level 1 to 3 at the $64\times$ downsampled resolution from all training cameras to add large Gaussians for level 4. A Gaussian splatted to any of the training cameras with a pixel coverage S_k smaller than S_T is considered too small, and is included for the next step of aggregation.

3. Theoretical Anti-aliasing Effectiveness of Gaussian Aggregation for 1D Signals

Our algorithm eschews low-pass filters for individual Gaussians as they do not mitigate the slow rendering speed. Instead, as shown in Fig. 1, we opt to substitute smaller Gaussians with fewer, larger ones, reducing the signal bandwidth and the number of primitives rendered. Heeding the reviewer's suggestion, we now delve deeper into the signal-processing analysis of our algorithm's anti-aliasing effect from first principles. Aliasing arises when a signal's bandwidth surpasses half the sampling frequency, as per Nyquist sampling theorem. Taking the mixture of 1D Gausssians $\Sigma_i e^{-a_i(x-x_i)^2}$ as an example, where $a_i = \frac{1}{2\sigma_i^2}$, we aim to prove that in our algorithm, they are consistently substituted with a Gaussian whose 3dB bandwidth is below the aliasing frequency threshold 0.5px^{-1}.

According to our algorithm, the mixture of Gaussians is first aggregated into an average Gaussian $g_{avg}(x) = e^{-a(x-\mu)^2}$ with average $a = \frac{1}{N}\Sigma_i a_i$ and $\mu = \frac{1}{N}\Sigma_i x_i$. We can apply the Fourier transform to convert it to the frequency domain to become $\mathcal{F}[g_{avg}(x)](f) = \sqrt{\frac{\pi}{a}} e^{-\pi^2 f^2/a}$. The 3dB bandwidth f_{3dB} is the frequency where the magnitude is $1/\sqrt{2}$ of its peak magnitude. By solving

$$\sqrt{\frac{\pi}{a}} e^{-\pi^2 f_{3dB}^2/a} = \frac{1}{\sqrt{2}}\sqrt{\frac{\pi}{a}}, \tag{1}$$

we find $f_{3dB} = \frac{1}{\pi}\sqrt{a \ln\sqrt{2}}$.

Our algorithm then scales standard deviation up by S_T/S, where $S_T = 2\text{px}$ is the selective rendering threshold and S is the pixel coverage of the Gaussian. We determine S by calculating the size at its level set, solving $e^{-a(0.5S)^2} = 1/N$ with $1/N = 1/255$ on 8-bit color images. This yields $S = 2\sqrt{\frac{1}{a}\ln N}$ and thus the scaled standard deviation becomes $\sigma' = \frac{2}{2\sqrt{\frac{1}{a}\ln N}}\sigma$. Given $a = \frac{1}{2\sigma^2}$, the scaled $a' = \frac{1}{a}\ln N \cdot a = \ln N$. Consequently, we calculate the 3dB bandwidth f'_{3dB} of the scaled Gaussian as:

$$\begin{aligned} f'_{3dB} &= \frac{1}{\pi}\sqrt{a' \ln\sqrt{2}} = \frac{1}{\pi}\sqrt{\ln N \cdot \ln\sqrt{2}} \\ &= 0.441\text{px}^{-1} < 0.5\text{px}^{-1}. \end{aligned} \tag{2}$$

This indicates that the bandwidth of the scaled Gaussians remains invariant to the attributes of the smaller Gaussians they replace, and is below half of the sampling frequency to avoid aliasing. While differing from the traditional low-pass filtering, our method is equally effective in anti-aliasing but more efficient in rendering.

Unbounded Scene Normalization. The Gaussians can be located at the range of $(-\infty, \infty)$ in unbounded scenes. This is not suitable for voxelization later as only a limited amount of voxels can be used. To normalize the unbounded space, the center region and the outer region are handled in different manners. The space bounded by a axis-aligned cube of length B defined by the span of all training cameras is considered the center region, and the rest is considered the outer region. To preserve the structure in the center region, the coordinates are linearly scaled from $[-B, B]$ to $[-1, 1]$. To normalize the unbounded outer region, the coordinates are non-linearly scaled from $(-\infty, \infty)$ to $(-2, 2)$.

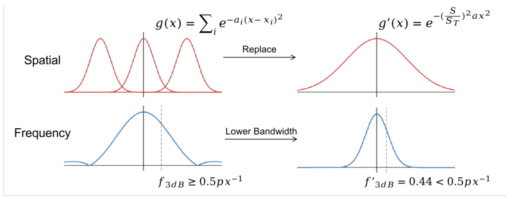

Figure 1. Instead of low-pass filters, we replace smaller Gaussians with fewer larger Gaussians, ensuring their bandwidth is below the aliasing threshold 0.5px^{-1}.

The exact normalization is as follows:

$$\mathbf{x}_{norm} = \begin{cases} \mathbf{x}/B, & \text{if } max(|\mathbf{x}|) \leq B \\ 2 - B/\mathbf{x}, & \text{otherwise} \end{cases}. \quad (3)$$

Voxelization. After the Gaussian positions are normalized to $[-2, 2]$, they need to be voxelized so that all Gaussians in one voxel are grouped together for the aggregation later. The size of the voxel increases as the resolutions decrease because coarser levels require fewer larger Gaussians. Specifically, when inserting large Gaussians for level l_m, the voxel size is chosen to be an empirical value of $(400/l_m)^3$. All Gaussians with their center in one voxel are grouped together for the next step. Although it is possible for a Gaussian to extent beyond the voxel while its center resides in the voxel, it is unlikely to reach too far as large Gaussians are filtered out in the earlier procedure.

Average Pooling and Enlargement After the small Gaussians are grouped in individual voxels, their parameters are averaged to create the large Gaussian. Specifically, the large Gaussian takes the average position, rotation, spherical harmonics features, opacity and scaling. However, a new Gaussian would be too small if it remains at this scaling. Consequently, we calculate the average pixel coverage of all the aggregated small Gaussians S_{avg} using their pixel coverage derived earlier. The scaling of the new Gaussian is then enlarged by S_T/S_{avg} for its pixel coverage to be approximately S_T, which is suitable to be rendered at level l_m. This average pooling is not perfect, but simple and effective enough to produce a reasonable initialization for the multi-scale training later.

4. Qualitative Ablation Study

To better compare the effectiveness of each of our proposed module qualitatively, we present the rendering results of our method and various ablation models in Fig. 2–6. The ablation model design follows the experiment section in the main paper. Specifically, the "+MS Train" model is trained using multi-scale images, but the Gaussians are only of a single scale as in 3D Gaussian Splatting [3]. The low-resolution performance is slightly improved, but the rendering speed is as slow as the original method. The "+Filter Small" model filters the small Gaussians based on the pixel coverage on top of the multi-scale training. It significantly accelerates the low-resolution rendering process, but the scene has some part missing as shown in the rendered images. The image rendered also has artifacts like black dots at low resolutions, caused by the filtered small Gaussians. The "+Insert Large" model inserts the large Gaussians from aggregation on top of the multi-scale training. It has good rendering speed and quality at low resolutions, but the image rendered at high resolution is over-smoothed. This is caused by the finer level Gaussians not filtered out but optimized together with the inserted large Gaussians at low resolutions. Our "Full Method" overcomes the weakness of the ablation models and produces high-quality rendering at fast speed on both high and low resolutions. The small Gaussians filtered improves the speed, and the large Gaussians inserted improves the quality at low resolutions. The qualitative ablation supports the effectiveness of our proposed components.

5. Quantitative Results on More Resolutions

We present the quantitative results of our method, the original 3D Gaussian Splatting[3], and the various ablation methods on more downsampled resolutions. The resolutions include those that are not used during training which demonstrate the performance and robustness of our model. The experiments are conducted on MipNeRF-360 dataset [1] as shown in Tab. 1, Tank and Temple dataset [4] as shown in Tab. 2, and Deep Blending dataset [2] as shown in Tab. 3.

6. Per-Scene Quantitative Results

We present the per-scene decomposition of the quantitative results of our method and the original 3D Gaussian splatting [3] in various resolutions. The experiments are carried on MipNeRF-360 dataset [1] as shown in Tab. 4, Tank and Temple dataset [4] as shown in Tab. 5, and Deep Blending dataset [2] as shown in Tab. 6. The scenes chosen to be tested on follow the experiments carried out in the original 3D Gaussian splatting paper [3].

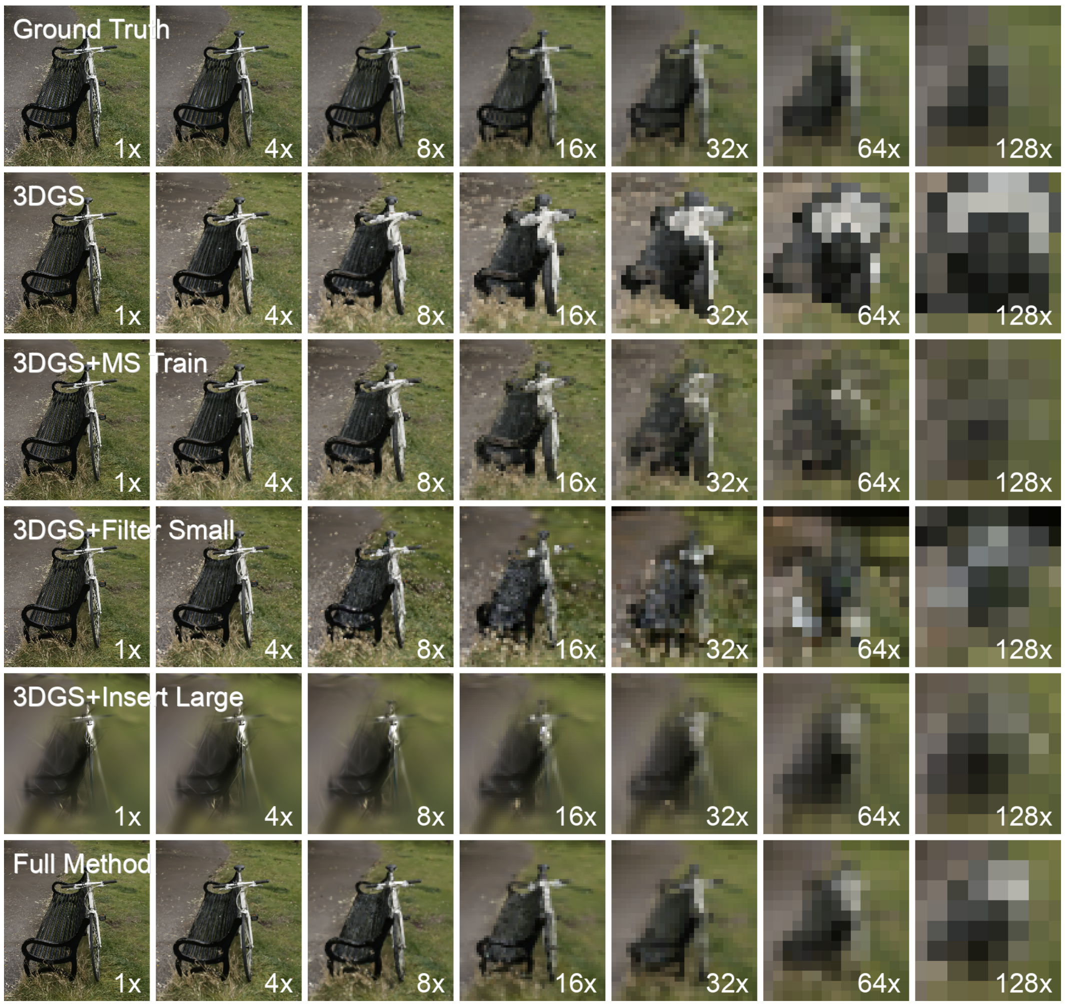

Figure 2. Qualitative ablation results of our proposed method on the "Bicycle" scene.

Table 1. Quantitative comparison and ablation study on MipNeRF 360 dataset [1] at more downsampled scales, with time in "ms".

Scale	1x			2x			4x			8x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	**27.52**	**0.142**	10.5	25.96	**0.124**	8.0	22.50	0.137	9.3	19.79	0.154	14.6
3DGS + MS Train	27.35	0.155	11.3	26.33	0.128	7.3	23.50	**0.126**	7.7	21.38	0.131	12.1
3DGS + Filter Small	27.40	0.153	10.0	26.42	0.129	6.8	23.81	0.149	5.4	21.73	0.175	5.1
3DGS + Insert Large	18.02	0.604	9.7	18.28	0.593	**3.4**	18.75	0.531	**2.5**	19.39	0.419	**2.2**
Our Method	27.39	0.155	**9.1**	**26.44**	0.134	6.3	**24.82**	0.132	5.4	**24.44**	**0.112**	5.1

Scale	16x			32x			64x			128x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	17.79	0.149	27.9	16.30	0.084	55.2	15.23	N.A.	103.3	14.55	N.A.	123.2
3DGS + MS Train	20.21	0.115	22.8	19.80	0.060	45.6	19.38	N.A.	84.8	18.75	N.A.	100.1
3DGS + Filter Small	20.02	0.186	4.8	18.81	0.090	**4.4**	17.38	N.A.	**4.6**	16.13	N.A.	**4.8**
3DGS + Insert Large	20.23	0.256	**2.7**	21.17	0.081	4.6	21.53	N.A.	7.1	20.25	N.A.	9.4
Our Method	**24.75**	**0.066**	4.9	**25.06**	**0.025**	4.7	**25.35**	N.A.	4.9	**22.55**	N.A.	5.0

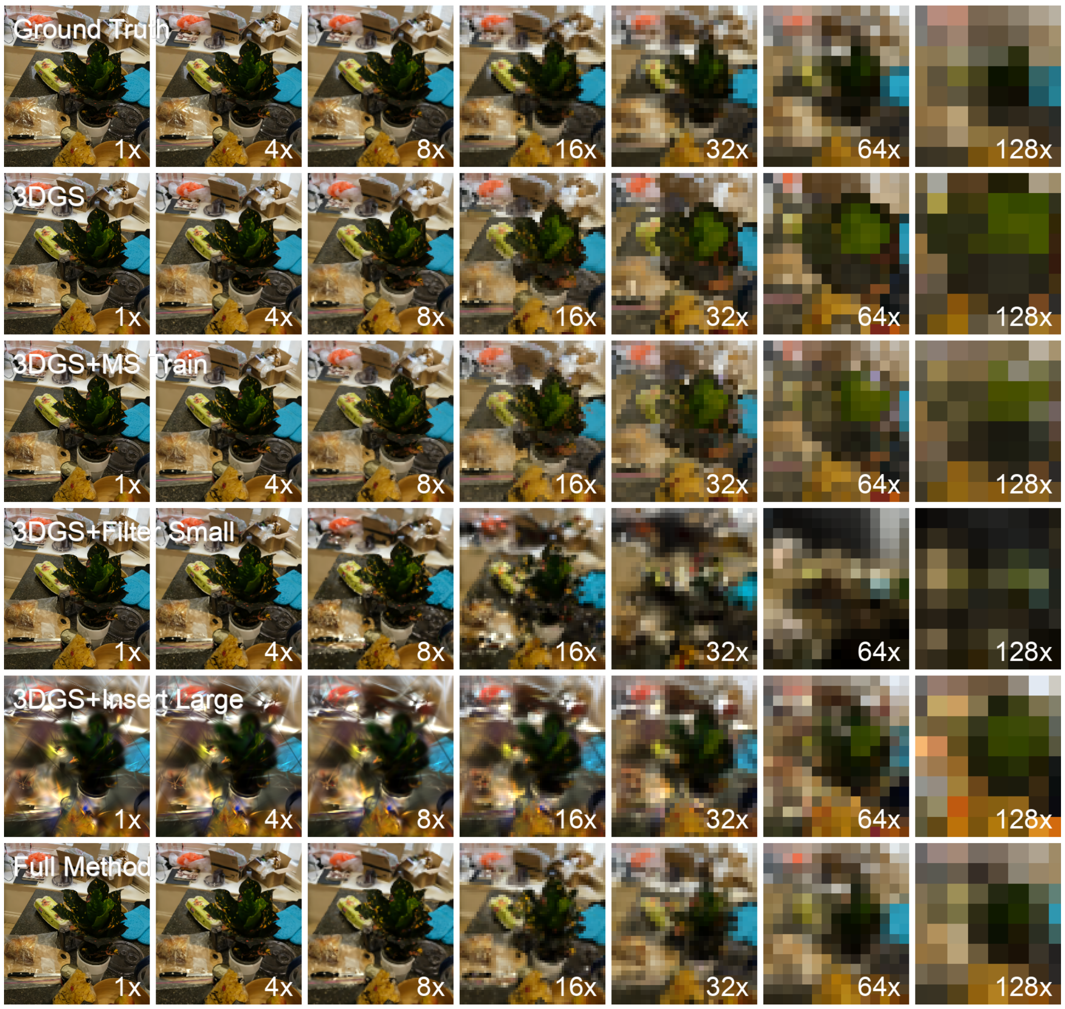

Figure 3. Qualitative ablation results of our proposed method on the "Counter" scene.

Table 2. Quantitative comparison and ablation study on Tank and Temple dataset [4] at more downsampled scales, with time in "ms".

Scale	1x			2x			4x			8x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	23.74	**0.096**	6.5	22.55	0.080	7.1	19.70	0.105	11.1	17.34	0.117	21.5
3DGS + MS Train	22.97	0.118	6.0	**23.04**	0.083	6.3	21.46	**0.086**	9.6	20.18	**0.080**	18.5
3DGS + Filter Small	**23.78**	0.100	5.6	22.76	**0.079**	5.1	20.12	0.107	4.5	18.62	0.122	4.4
3DGS + Insert Large	10.84	0.697	**5.1**	10.96	0.719	**2.4**	11.15	0.703	**1.7**	11.40	0.631	**1.6**
Our Method	23.46	0.111	7.6	22.44	0.095	5.6	**21.92**	0.087	4.7	**20.88**	0.082	4.6

Scale	16x			32x			64x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	15.61	0.068	43.4	14.45	N.A.	70.9	13.88	N.A.	82.6
3DGS + MS Train	18.56	0.049	37.4	17.41	N.A.	61.7	16.54	N.A.	71.7
3DGS + Filter Small	17.41	0.072	4.4	16.05	N.A.	4.5	14.95	N.A.	4.7
3DGS + Insert Large	11.73	0.447	**1.7**	12.14	N.A.	**2.1**	12.62	N.A.	**2.5**
Our Method	**20.91**	**0.034**	4.8	**21.01**	N.A.	5.4	**19.67**	N.A.	5.9

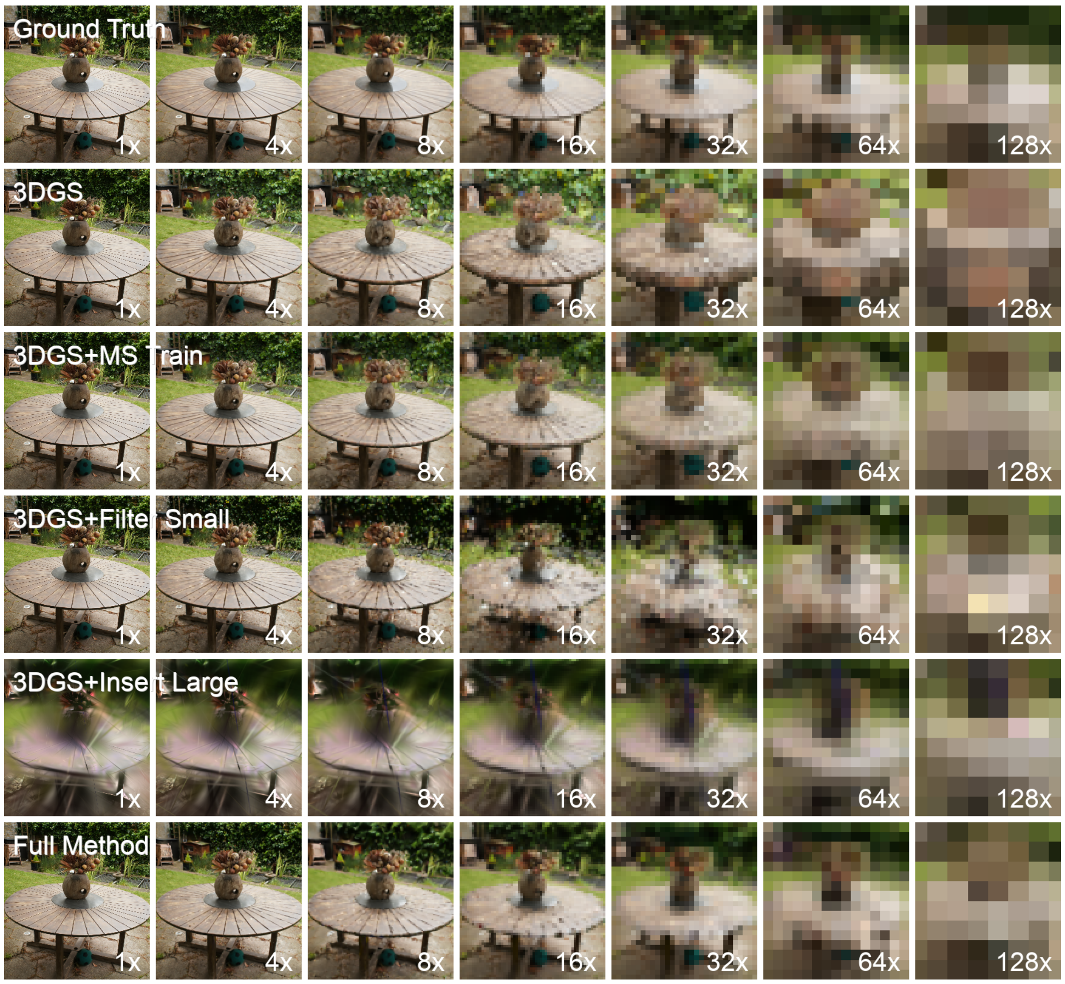

Figure 4. Qualitative ablation results of our proposed method on the "Garden" scene.

Table 3. Quantitative comparison and ablation study on Deep Blending dataset [2] at more downsampled scales, with time in "ms".

Scale	1x			2x			4x			8x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	29.65	**0.094**	8.6	29.41	0.065	6.6	27.48	0.066	7.5	24.67	0.076	11.3
3DGS + MS Train	29.46	0.102	6.6	29.42	0.069	4.8	28.18	**0.062**	5.3	26.15	0.065	8.0
3DGS + Filter Small	29.68	0.095	6.7	29.53	**0.064**	4.9	28.26	0.064	4.2	26.51	0.082	3.8
3DGS + Insert Large	20.59	0.379	**4.6**	20.67	0.381	**2.2**	20.83	0.336	**1.6**	21.07	0.263	**1.7**
Our Method	**29.70**	0.096	7.4	**29.58**	0.065	4.8	**28.43**	0.064	3.9	**27.59**	**0.063**	3.6

Scale	16x			32x			64x		
Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
3D Gaussian[3]	22.06	0.067	20.7	19.74	N.A.	36.3	17.75	N.A.	59.7
3DGS + MS Train	24.13	0.055	14.3	22.09	N.A.	24.8	20.03	N.A.	41.3
3DGS + Filter Small	24.52	0.078	3.6	22.01	N.A.	3.3	18.29	N.A.	**3.2**
3DGS + Insert Large	21.29	0.143	**2.1**	21.14	N.A.	**2.8**	20.10	N.A.	4.2
Our Method	**27.66**	**0.036**	3.4	**27.22**	N.A.	3.3	**25.70**	N.A.	3.4

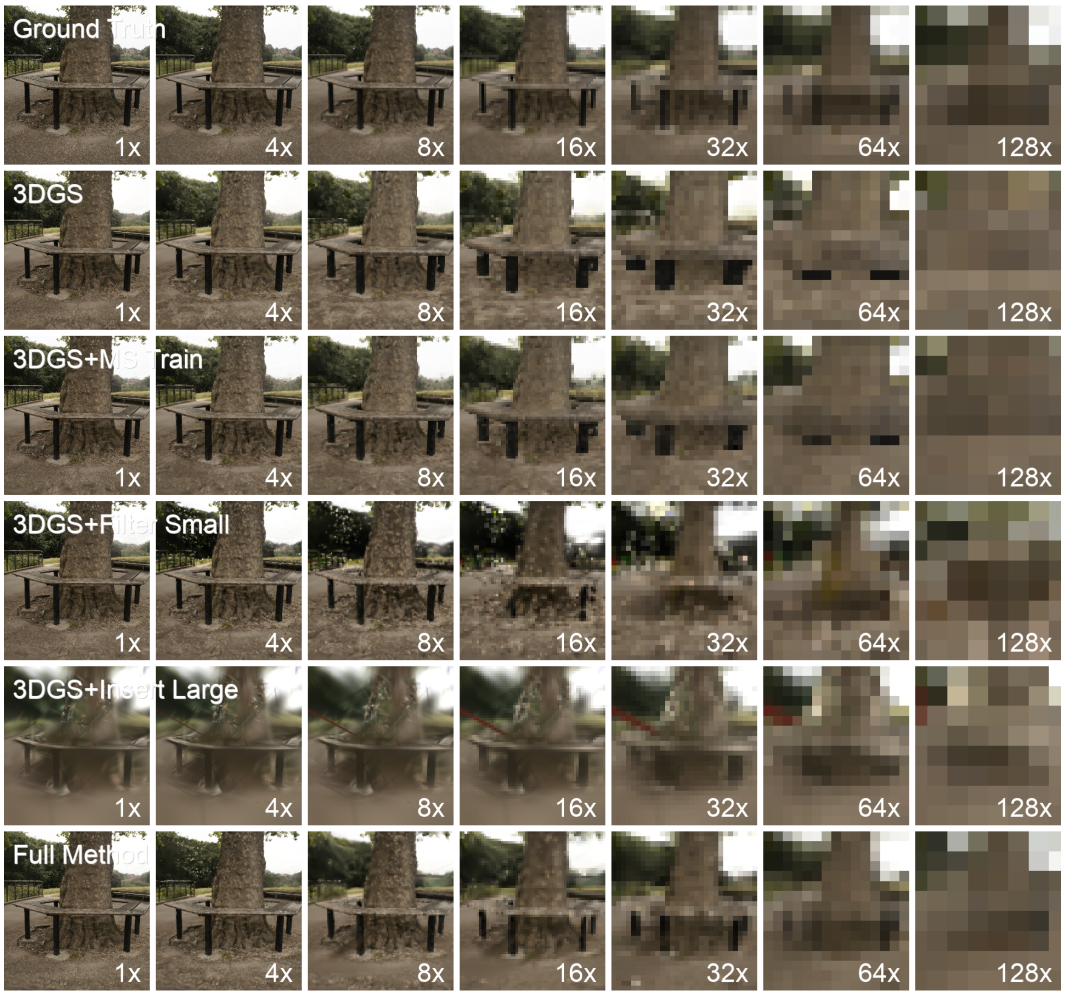

Figure 5. Qualitative ablation results of our proposed method on the ”Treehill” scene.

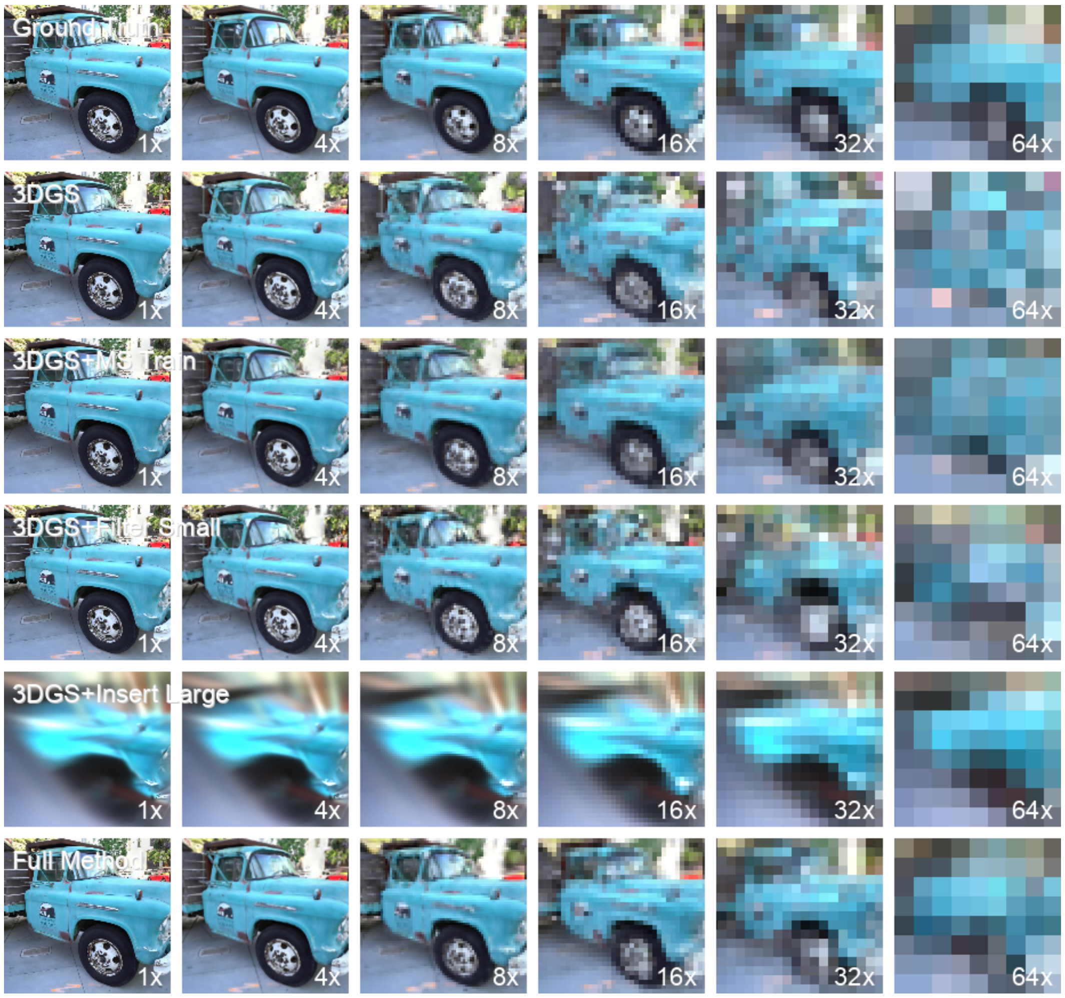

Figure 6. Qualitative ablation results of our proposed method on the ”Truck” scene.

Table 4. Per-scene performance decomposition on MipNeRF-360 dataset[1]. Time measured in 'ms'.

	Scale	1x			4x			16x			64x			128x		
Scene	Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
garden	3D-GS[3]	**27.27**	**0.070**	15.0	20.42	0.136	14.4	16.74	0.166	48.8	14.92	N.A.	200.9	14.29	N.A.	245.0
garden	Ours	27.16	0.080	**11.8**	**23.99**	**0.112**	**7.8**	**26.41**	**0.044**	**7.5**	**24.79**	N.A.	**8.6**	**21.19**	N.A.	**9.6**
flowers	3D-GS[3]	**21.41**	**0.309**	9.1	18.89	0.239	8.8	15.46	0.165	24.9	13.90	N.A.	93.2	13.69	N.A.	112.2
flowers	Ours	21.11	0.333	**8.1**	**20.83**	**0.234**	**5.7**	**21.97**	**0.093**	**5.1**	**22.69**	N.A.	**4.9**	**21.82**	N.A.	**5.0**
treehill	3D-GS[3]	22.60	**0.274**	10.0	21.63	**0.232**	9.7	18.71	0.193	24.6	16.19	N.A.	90.6	15.52	N.A.	97.0
treehill	Ours	**22.64**	0.291	**8.7**	**22.31**	0.239	**5.8**	**23.55**	**0.072**	**5.4**	**24.28**	N.A.	**4.9**	**22.27**	N.A.	**5.0**
bicycle	3D-GS[3]	**25.15**	**0.164**	18.8	19.71	0.178	15.5	16.27	0.215	43.9	14.99	N.A.	163.8	15.15	N.A.	187.0
bicycle	Ours	24.44	0.210	**13.4**	**24.76**	**0.131**	**7.4**	**25.00**	**0.081**	**6.4**	**26.02**	N.A.	**6.5**	**21.56**	N.A.	**6.9**
counter	3D-GS[3]	29.15	**0.099**	7.5	24.81	0.084	6.4	17.94	0.101	19.2	14.32	N.A.	60.4	13.39	N.A.	74.6
counter	Ours	**29.17**	0.100	**6.6**	**26.77**	**0.076**	**3.3**	**23.44**	**0.057**	**2.8**	**24.59**	N.A.	**2.7**	**21.14**	N.A.	**2.7**
kitchen	3D-GS[3]	**31.70**	**0.064**	9.3	23.95	**0.081**	8.5	18.50	0.093	35.4	15.00	N.A.	124.4	14.15	N.A.	150.3
kitchen	Ours	31.64	0.064	**8.1**	**25.93**	0.089	**4.2**	**24.16**	**0.049**	**3.9**	**25.35**	N.A.	**3.3**	**21.50**	N.A.	**3.2**
room	3D-GS[3]	**31.63**	**0.093**	8.0	26.60	0.057	5.1	19.50	0.096	12.0	15.50	N.A.	49.2	14.37	N.A.	70.8
room	Ours	31.51	0.094	**6.6**	**28.95**	**0.053**	**3.1**	**28.15**	**0.025**	**2.9**	**25.77**	N.A.	**2.9**	**21.82**	N.A.	**2.9**
stump	3D-GS[3]	**26.75**	**0.138**	**10.6**	22.24	0.152	10.1	18.57	0.188	26.5	17.33	N.A.	95.2	16.97	N.A.	114.0
stump	Ours	26.59	0.152	12.9	**23.52**	**0.150**	**8.2**	**25.22**	**0.112**	**7.2**	**29.22**	N.A.	**7.1**	**29.09**	N.A.	**7.2**
bonsai	3D-GS[3]	32.04	**0.065**	6.0	24.23	**0.075**	5.3	18.43	0.126	15.4	14.95	N.A.	52.4	13.46	N.A.	57.9
bonsai	Ours	**32.27**	0.067	**5.5**	**26.32**	0.106	**3.3**	**24.87**	**0.062**	**2.8**	**25.40**	N.A.	**2.9**	**22.53**	N.A.	**2.8**

Table 5. Per-scene performance decomposition on Tank and Temple dataset[4]. Time measured in 'ms'.

	Scale	1x			4x			16x			64x		
Scene	Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
truck	3D-GS[3]	**25.39**	**0.064**	**7.3**	19.97	0.103	11.3	15.69	0.064	49.2	14.20	N.A.	89.1
truck	Ours	24.94	0.078	9.0	**23.67**	**0.059**	**5.4**	**22.62**	**0.024**	**6.0**	**19.99**	N.A.	**8.6**
train	3D-GS[3]	**22.09**	**0.129**	**5.8**	19.42	**0.108**	10.9	15.54	0.072	37.6	13.57	N.A.	76.1
train	Ours	21.98	0.144	6.2	**20.17**	0.114	**3.9**	**19.21**	**0.044**	**3.5**	**19.36**	N.A.	**3.3**

Table 6. Per-scene performance decomposition on Deep Blending dataset[2] Time measured in 'ms'.

	Scale	1x			4x			16x			64x		
Scene	Metric	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓	PSNR↑	LPIPS↓	Time↓
drjohnson	3D-GS[3]	29.14	**0.106**	10.1	27.23	0.079	9.3	22.73	0.078	26.3	18.60	N.A.	67.6
drjohnson	Ours	**29.19**	0.108	**8.6**	**27.96**	**0.078**	**4.4**	**26.80**	**0.051**	**3.9**	**27.19**	N.A.	**3.8**
playroom	3D-GS[3]	30.15	**0.082**	7.0	27.72	0.053	5.7	21.40	0.056	15.0	16.89	N.A.	51.8
playroom	Ours	**30.20**	0.084	**6.2**	**28.89**	**0.051**	**3.4**	**28.53**	**0.020**	**3.0**	**24.22**	N.A.	**3.0**

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 3, 8

[2] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), 2018. 2, 5, 8

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 2, 3, 4, 5, 8

[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. 2, 4, 8