# Lightweight Gradient-Aware Upscaling of 3D Gaussian Splatting Images

## Supplementary Material

### A. Qualitative Results for Different Upsampling Rates

Our method enables image upscaling with arbitrary factors, including fractional values, offering flexibility beyond fixed integer magnifications. While our paper primarily showcases visual results for higher upscaling factors (4x,8x), our approach also demonstrates superior performance at lower magnifications, such as 2× or 3×, when compared to traditional interpolation techniques like bicubic or Lanczos (see Fig. 4). Specifically, our method reduces artifacts and preserves finer details more effectively.

However, these improvements are most apparent when the upscaled image is displayed at its native resolution, where the screen's pixel grid aligns with the image pixels. When viewed in formats like PDFs, where rendering software may resample, process, or scale images dynamically, these advantages can become less noticeable due to unintended alterations introduced by the viewer. In Fig. 4, we show qualitative results for 3x upscaling.

### B. Performance Analysis

In this section, we provide a more detailed performance analysis of our approach. Tab. 2 presents rendering times for different upscaling methods and upscaling factors, highlighting the computational cost of each configuration.

Additionally, we analyze the performance during training, focusing on the timing of key operations. As shown in Fig. 1 we measure the time required for the forward and backward pass of the most computationally expensive functions: rendering, SSIM computation, and upscaling. Our results indicate that the gradient computation has a negligible impact on the performance of the render function.

However, computing SSIM at a higher resolution (4×) is significantly more expensive. Furthermore, the backward pass of our upscaling function is relatively slow and constitutes a major bottleneck. This is particularly evident in our spline upscaler, which relies on a naive implementation that extensively uses atomic operations in the backward pass. Using subgroup operations and more efficient synchronization methods could significantly improve performance.

### C. Comparison to Mip Splatting

We also evaluated Mip-Splatting [9] in comparison to our approach. Mip Splatting introduces a 3D filter that accounts for the scene's sampling rate (training image resolution). By dilating the Gaussians during the reconstruction based on the sampling rate, Mip Splatting effectively reduces straw effects that can otherwise be seen in 3DGS reconstruction.

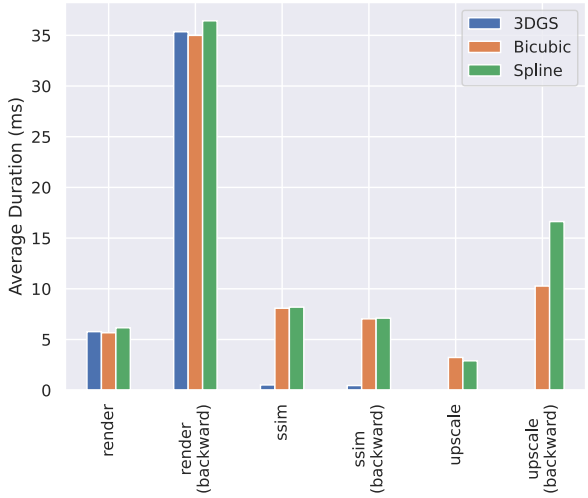



Figure 1. Averge time of different operations in the training pipeline. The MipNeRF360 garden scene with 5 Million Gaussians and the full resolution was used to obtain measurements. 3DGS was trained at $\frac{1}{4}$ resolution while Spline and Bicubic were trained with 4x upscaling.

However, this filtering process enlarges the Gaussians and decreases their opacity. This affects the rendering process in two key ways. First, the increased Gaussian size results in more Gaussians contributing to each pixel, thereby increasing rendering and training time. Second, the reduced opacity diminishes the effectiveness of early termination in the rendering pipeline. Since early termination typically halts processing more Gaussians when a pixel's alpha value nears one, lower opacity Gaussians require more blending steps before reaching full coverage, thereby increasing the overall rendering and training time (see Tab. 1).

While Mip Splatting improves image consistency at high resolutions, its computational overhead makes it less suitable for real-time applications on low-end devices. Nevertheless, our upscaling method can be applied to scenes reconstructed with Mip Splatting, thereby speeding up the rendering process.

### D. Lanczos Image Interpolation

We compare Lanczos and bicubic interpolation for image upscaling. Lanczos, though theoretically superior in edge preservation [2], showed no major improvements over bicubic in our experiments. Additionally, it is more computationally expensive to compute.

Given the lack of significant visual benefits, bicubic remains the preferable choice as the baseline in our experi-

| Scale | Method | SSIM ↑ | PSNR ↑ | LPIPS ↓ | FPS ↓ | Train ↓ |
|---|---|---|---|---|---|---|
| 2 | 3DGS | 0.809 | 26.97 | 0.308 | 15 | 105 |
| | Mip | 0.815 | 27.23 | 0.302 | 10 | 143 |
| | Spline (Ours) | 0.813 | 27.00 | 0.31 | 41 | 139 |
| 4 | 3DGS | 0.783 | 26.35 | 0.347 | 15 | 59 |
| | Mip | 0.806 | 27.14 | 0.317 | 10 | 73 |
| | Spline (Ours) | 0.809 | 26.85 | 0.322 | 90 | 91 |
| 8 | 3DGS | 0.681 | 24.67 | 0.446 | 14 | 48 |
| | Mip | 0.775 | 26.37 | 0.371 | 8 | 55 |
| | Spline (Ours) | 0.776 | 25.95 | 0.385 | 122 | 80 |

Table 1. Metrics for Mip-Splatting [9] compared to baseline 3DGS [5] and our Spline Upscaler. Average for all scenes in Mip-Nerf360 [1] is reported. Metrics are evaluated at full resolution (5187x3361 pixels). 3DGS and Mip were trained at lower resolution (scale) while Spline (ours) uses upscaling to match ground truth image resolution during training.

ments. Visual comparisons can be found in Figure Fig. 2.

## E. Detailed Metrics

We provided detailed metrics/results for all the experiments in the paper. Tabs. 3 to 5 show detailed results for training with upscaling for each scene from the different datasets used in the paper. Additionally, in Fig. 4, we provide additional visual results for the scenes not shown in the paper.

| Method | Scale | Render | Upscale | Speedup |
|---|---|---|---|---|
| 3DGS | 1 | 72.4 ms | - | - |
| Bicubic | 2 | 24.6 ms | 2.0 ms | ×2.7 ↑ |
| | 3 | 15.4 ms | 1.8 ms | ×4.2 ↑ |
| | 4 | 11.8 ms | 1.8 ms | ×5.3 ↑ |
| | 8 | 10.5 ms | 1.6 ms | ×6.0 ↑ |
| DLSS | 2 | 24.0 ms | 8.9 ms | ×2.2 ↑ |
| | 3 | 15.1 ms | 8.0 ms | ×3.1 ↑ |
| NinaSR-B1 | 2 | 23.8 ms | 604.8 ms | ×0.1 ↓ |
| | 3 | 15.0 ms | 275.3 ms | ×0.2 ↓ |
| | 4 | 11.7 ms | 158.9 ms | ×0.4 ↓ |
| | 8 | 10.7 ms | 45.4 ms | ×1.3 ↑ |
| Spline (Ours) | 2 | 24.7 ms | 4.0 ms | ×2.5 ↑ |
| | 3 | 15.6 ms | 1.8 ms | ×4.2 ↑ |
| | 4 | 11.9 ms | 2.2 ms | ×5.1 ↑ |
| | 8 | 10.7 ms | 1.7 ms | ×5.9 ↑ |

Table 2. Render times for different methods and upscaling factors. The target resolution is 5187x3361. The garden scene with 5 Million Gaussians is used. The cameras from the test set are used, and measurements are averaged over all images.

## F. DL-based Upscalers

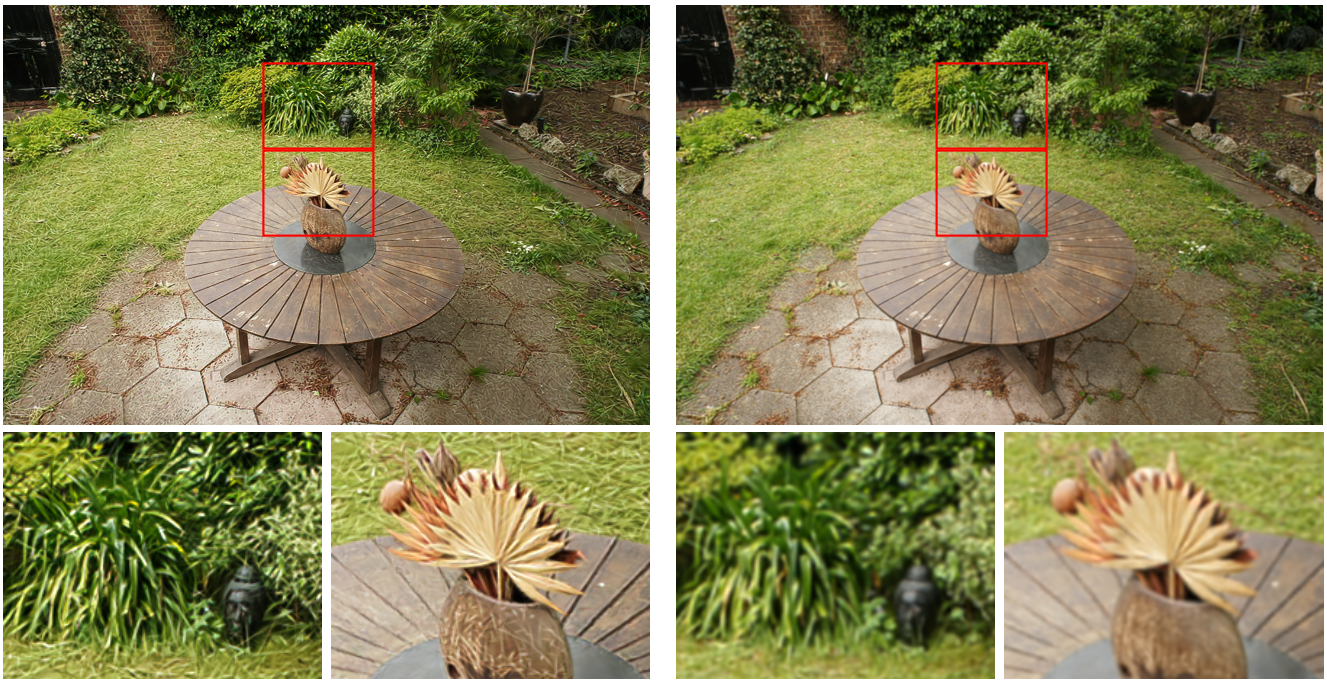
Sec. 6.3 of the main manuscript discusses the results achieved via DL-based upscaling techniques. DL-based upscalers can be compared to our proposed spline-based upscaling technique with respect to upscaling quality, as in Fig. 6 of the main manuscript. Furthermore, a DL-based upscaler can be incorporated into the training pipeline for low-resolution renderings, as illustrated in Fig. 4 of the main manuscript. In this section, we elaborate on this incorporation.

We considered many pre-trained DL-based upscalers from the torchSR project [4], of which EDSR [6] is supposed to achieve the best results. However, incorporating this network into the 3DGS training pipeline turned out to be impossible, as the model ran out of memory on the NVIDIA L4 GPU with 24GB of video memory. Therefore, we decided to use the smaller DL-based upscale NinaSR-B1 [3], which the torchSR project recommends for "practical applications" [4].

However, when training models with low-resolution renderings upscaled to the ground truth image resolution via the pre-trained NinaSR-B1, we could not achieve high-quality results. Firstly, even with NinaSR-B1 we were running out of memory when training with ground truth images of resolution $2594 \times 1681$ (called images_2 in the dataset) for the Garden scene from Mip-NeRF [1]. Consequently, we decided to use an upscaled resolution of $1296 \times 840$ pixels (images_4). Qualitative and quantitative results can be seen in Fig. 3. The deep learning-based upscaler produces overly sharp images that are not consistent across different views, leading to an overall worse result. Our spline-based approach is able to achieve significantly higher PSNR values (cf. Fig. 3).

(a) Bicubic (b) Lanczos (c) Spline (Ours)

Figure 2. Comparison of $4\times$ upscaling for different upscaling methods. Bicubic and Lanczos observe staircase and rining artifacts which are not present in Spline upscaling.



(a) NinaSR-B1 $3\times$, PSNR = 21.61, 128min (train) (b) Spline $3\times$ (ours), PSNR = 25.30, 51min (train)

Figure 3. Comparison of training results for the Garden scene from Mip-NeRF [1]. Training image resolutions of $1296 \times 840$ were used.

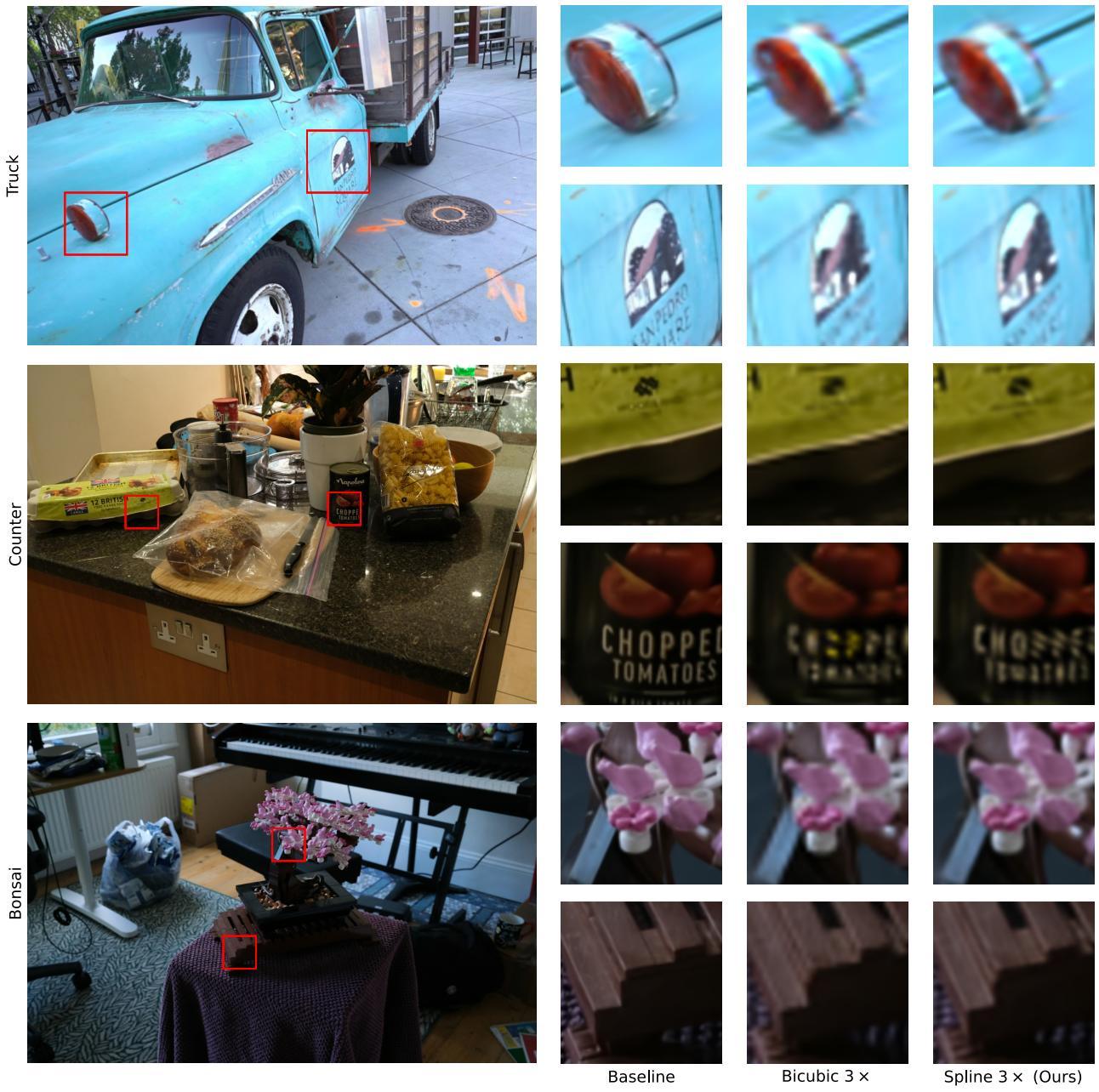



Figure 4. Comparison of image quality: (Left) Ground Truth, (Baseline) full resolution training with full-resolution rendering, (Bicubic) $3\times$ upscaling during training with bicubic interpolation, (Spline) $3\times$ upscaling during training with spline interpolation (ours).

| Dataset | Scene | 3DGS | | | | Bicubic | | | | Spline (Ours) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration |
| MipNeRF360 | bicycle | 24.402 | 0.733 | 0.338 | 116 | 24.434 | 0.737 | 0.340 | 141 | 24.397 | 0.734 | 0.344 | 162 |
| | bonsai | 32.021 | 0.940 | 0.267 | 76 | 31.520 | 0.937 | 0.270 | 87 | 32.071 | 0.940 | 0.268 | 95 |
| | counter | 28.507 | 0.912 | 0.255 | 89 | 28.374 | 0.915 | 0.254 | 99 | 28.391 | 0.915 | 0.253 | 107 |
| | flowers | 20.871 | 0.598 | 0.414 | 119 | 20.984 | 0.604 | 0.418 | 145 | 20.955 | 0.603 | 0.420 | 165 |
| | garden | 26.351 | 0.794 | 0.261 | 130 | 26.281 | 0.800 | 0.260 | 154 | 26.330 | 0.799 | 0.263 | 178 |
| | kitchen | 31.277 | 0.920 | 0.186 | 90 | 31.093 | 0.922 | 0.188 | 99 | 31.022 | 0.923 | 0.185 | 109 |
| | room | 31.549 | 0.921 | 0.269 | 82 | 31.585 | 0.923 | 0.268 | 93 | 31.657 | 0.923 | 0.266 | 101 |
| | stump | 26.248 | 0.797 | 0.366 | 114 | 26.446 | 0.804 | 0.364 | 137 | 26.442 | 0.803 | 0.365 | 160 |
| | treehill | 21.490 | 0.669 | 0.418 | 126 | 21.788 | 0.679 | 0.425 | 150 | 21.732 | 0.678 | 0.427 | 173 |
| Deep Blending | drjohnson | 29.259 | 0.898 | 0.251 | 17 | 29.258 | 0.899 | 0.251 | 18 | 29.120 | 0.901 | 0.249 | 19 |
| | playroom | 30.029 | 0.904 | 0.243 | 17 | 30.090 | 0.905 | 0.246 | 17 | 30.266 | 0.909 | 0.242 | 18 |
| Tanks and Temples | train | 19.966 | 0.754 | 0.254 | 18 | 20.201 | 0.761 | 0.265 | 19 | 20.622 | 0.788 | 0.235 | 19 |
| | truck | 24.219 | 0.844 | 0.185 | 16 | 24.173 | 0.837 | 0.211 | 17 | 24.652 | 0.864 | 0.177 | 17 |

Table 3. Training with $2\times$ upscaling at full resolution compared to 3DGS with no upscaling in training. Duration is reported in minutes.

| Dataset | Scene | 3DGS | | | | Bicubic | | | | Spline (Ours) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration |
| MipNeRF360 | bicycle | 24.216 | 0.708 | 0.375 | 60 | 24.222 | 0.722 | 0.365 | 95 | 24.317 | 0.729 | 0.352 | 105 |
| | bonsai | 30.728 | 0.915 | 0.310 | 50 | 31.053 | 0.925 | 0.293 | 63 | 31.110 | 0.934 | 0.282 | 68 |
| | counter | 27.082 | 0.875 | 0.298 | 54 | 27.906 | 0.899 | 0.291 | 68 | 28.162 | 0.909 | 0.269 | 72 |
| | flowers | 20.630 | 0.571 | 0.440 | 63 | 21.010 | 0.598 | 0.433 | 95 | 20.984 | 0.601 | 0.428 | 106 |
| | garden | 25.834 | 0.755 | 0.322 | 67 | 25.916 | 0.767 | 0.308 | 100 | 26.192 | 0.786 | 0.282 | 110 |
| | kitchen | 30.194 | 0.886 | 0.246 | 56 | 30.258 | 0.900 | 0.237 | 70 | 31.262 | 0.920 | 0.199 | 73 |
| | room | 31.003 | 0.905 | 0.293 | 52 | 30.933 | 0.909 | 0.303 | 65 | 31.349 | 0.918 | 0.283 | 68 |
| | stump | 26.050 | 0.780 | 0.389 | 60 | 26.318 | 0.800 | 0.372 | 93 | 26.336 | 0.801 | 0.367 | 104 |
| | treehill | 21.400 | 0.646 | 0.447 | 65 | 21.801 | 0.673 | 0.444 | 99 | 21.941 | 0.680 | 0.435 | 110 |
| Deep Blending | drjohnson | 28.596 | 0.873 | 0.286 | 14 | 28.433 | 0.871 | 0.310 | 15 | 28.750 | 0.887 | 0.286 | 16 |
| | playroom | 29.180 | 0.880 | 0.285 | 14 | 28.958 | 0.880 | 0.313 | 15 | 29.424 | 0.893 | 0.286 | 15 |
| Tanks and Temples | train | 18.944 | 0.651 | 0.356 | 21 | 18.457 | 0.636 | 0.412 | 21 | 19.598 | 0.708 | 0.341 | 22 |
| | truck | 21.684 | 0.703 | 0.325 | 17 | 22.096 | 0.719 | 0.372 | 18 | 23.002 | 0.780 | 0.305 | 17 |

Table 4. Training with $4\times$ upscaling at full resolution compared to 3DGS with no upscaling in training. Duration is reported in minutes.

| Dataset | Scene | 3DGS | | | | Bicubic | | | | Spline (Ours) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration | PSNR | SSIM | LPIPS | Duration |
| MipNeRF360 | bicycle | 22.941 | 0.601 | 0.467 | 47 | 23.008 | 0.657 | 0.479 | 88 | 23.456 | 0.683 | 0.430 | 90 |
| | bonsai | 28.071 | 0.822 | 0.418 | 47 | 29.310 | 0.885 | 0.370 | 63 | 29.915 | 0.911 | 0.322 | 64 |
| | counter | 25.848 | 0.774 | 0.407 | 49 | 26.451 | 0.851 | 0.394 | 67 | 27.609 | 0.882 | 0.333 | 66 |
| | flowers | 19.658 | 0.473 | 0.500 | 48 | 20.641 | 0.563 | 0.498 | 90 | 20.697 | 0.578 | 0.475 | 92 |
| | garden | 23.861 | 0.621 | 0.438 | 50 | 24.276 | 0.665 | 0.445 | 94 | 25.064 | 0.712 | 0.383 | 96 |
| | kitchen | 27.245 | 0.753 | 0.413 | 50 | 27.524 | 0.802 | 0.385 | 68 | 29.168 | 0.871 | 0.287 | 68 |
| | room | 29.320 | 0.854 | 0.366 | 45 | 29.427 | 0.878 | 0.378 | 62 | 29.785 | 0.895 | 0.336 | 62 |
| | stump | 24.567 | 0.666 | 0.488 | 46 | 25.496 | 0.764 | 0.455 | 88 | 25.934 | 0.786 | 0.413 | 90 |
| | treehill | 20.532 | 0.561 | 0.516 | 49 | 21.652 | 0.644 | 0.515 | 91 | 21.907 | 0.666 | 0.488 | 92 |

Table 5. Training with $8\times$ upscaling at full resolution compared to 3DGS with no upscaling in training. Duration is reported in minutes.

## G. Spline Image Interpolation

A bicubic spline can be parameterized with a third-order polynomial with the coefficients $A \in \mathbb{R}^{4\times4}$:

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j \tag{1}$$

The partial derivatives of the spline are given by:

$$\frac{\partial p(x,y)}{\partial x} = \sum_{i=1}^{3}\sum_{j=0}^{3} i a_{ij}x^{i-1} y^j \tag{2}$$

$$\frac{\partial p(x,y)}{\partial y} = \sum_{i=0}^{3}\sum_{j=1}^{3} j a_{ij}x^{i} y^{j-1} \tag{3}$$

$$\frac{\partial^2 p(x,y)}{\partial x \partial y} = \sum_{i=1}^{3}\sum_{j=1}^{3} ij a_{ij}x^{i-1} y^{j-1} \tag{4}$$

The corner values $f(0,0), f(1,0), f(0,1), f(1,1)$ and their derivatives, e.g., $\frac{\partial f(0,0)}{\partial x}, \frac{\partial f(0,0)}{\partial y}, \frac{\partial f(0,0)}{\partial x \partial y}$ are known. The value of the spline and its derivatives must be equal to $f(x,y)$ at the corner points.

The problem of calculating the coefficients $A$ can be formulated as a linear problem:

$$F = CAC^T \tag{5}$$

$$A = C^{-1}F(C^T)^{-1} \tag{6}$$

$$F = \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix}. \tag{7}$$

With $C$ being the coefficients of the cubic spline at the respective points:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \tag{8}$$

The coefficients are derived from a cubic function and its derivative at 0 and 1:

$$f(x) = a_0 \quad +a_1 x \quad +a_2 x^2 \quad +a_3 x^3 \tag{9}$$

$$f_x(x) = \quad +a_1 \quad +2a_2 x \quad +3a_3 x^2 \tag{10}$$

$$f(0) = 1a_0 \quad +0 \quad +0 \quad +0 \tag{11}$$

$$f(1) = 1a_0 \quad +1a_1 \quad +1a_2 \quad +1a_3 \tag{12}$$

$$f_x(0) = 0 \quad +1a_1 \quad +0 \quad +0 \tag{13}$$

$$f_x(1) = 0 \quad +1a_1 \quad +2a_2 \quad +3a_3 \tag{14}$$

We refer to [7] for a more detailed explanation.

# 1. 3DGS Gradient Computation

In the following, we describe how the screen space gradients are computed in the forward pass, and we dicsuss the backward pass required for training with differentiable image upscaling.

## A. Forward pass

With a 3DGS model comprising $N$ Gaussians, the image $I(x,y)$ is rendered as follows:

$$I(x,y) = \sum_{i=1}^{N} T_i(x,y)\alpha_i(x,y)c_i \tag{15}$$

$$\alpha_i(x,y) = \sigma_i \exp(g_i(x,y)) \tag{16}$$

$$g_i(x,y) = -d_i \Sigma_i d_i^T \tag{17}$$

$$d_i = \begin{bmatrix} x-\mu_x & y-\mu_y \end{bmatrix} \tag{18}$$

$$T_i(x,y) = \prod_{j=1}^{i-1}(1-\alpha_j(x,y)) \tag{19}$$

Here, $c_i$ is the color of the i-th Gaussian and $\sigma_i$ its opacity. $\Sigma_i \in \mathbb{R}^{2\times2}$ is the 2D covariance matrix calculated with EWA [10] splatting for each Gaussian.

The image computation can be reformulated using front-to-back $\alpha$-blending:

$$I(x,y) = B_N \tag{20}$$

$$B_i = B_{i-1} + (1-A_{i-1})\alpha_i c_i \tag{21}$$

$$A_i = A_{i-1} + \alpha_i(1-A_{i-1}) \tag{22}$$

$$B_0 = 0 \tag{23}$$

$$A_0 = 0 \tag{24}$$

Using this formulation, the partial derivatives with respect to $x$ and $y$ can be computed as

$$\frac{\partial I(x,y)}{\partial x} = \frac{\partial B_N}{\partial x} \tag{25}$$

$$\frac{\partial B_i}{\partial x} = \frac{\partial B_{i-1}}{\partial x} + c_i((1-A_{i-1})\frac{\partial \alpha_i}{\partial x} - \frac{\partial A_{i-1}}{\partial x}\alpha_i) \tag{26}$$

$$\frac{\partial B_i}{\partial y} = \frac{\partial B_{i-1}}{\partial y} + c_i((1-A_{i-1})\frac{\partial \alpha_i}{\partial y} - \frac{\partial A_{i-1}}{\partial y}\alpha_i) \tag{27}$$

$$\frac{\partial^2 B_i}{\partial x \partial y} = \frac{\partial^2 B_{i-1}}{\partial x \partial y} + c_i((1-A_{i-1})\frac{\partial^2 \alpha_i}{\partial x \partial y} - \frac{\partial A_{i-1}}{\partial y}\frac{\partial \alpha_i}{\partial x} \tag{28}$$

$$- \frac{\partial^2 A_{i-1}}{\partial x \partial y}\alpha_i - \frac{\partial A_{i-1}}{\partial x}\frac{\partial \alpha_i}{\partial y}) \tag{29}$$

The partial derivatives of $A_i$ are

$$\frac{\partial A_i}{\partial x} = \frac{\partial A_{i-1}}{\partial x}(1-\alpha_i) + (1-A_{i-1})\frac{\partial \alpha_i}{\partial x} \tag{30}$$

$$\frac{\partial A_i}{\partial y} = \frac{\partial A_{i-1}}{\partial y}(1-\alpha_i) + (1-A_{i-1})\frac{\partial \alpha_i}{\partial y} \tag{31}$$

$$\frac{\partial^2 A_i}{\partial x \partial y} = \frac{\partial^2 A_{i-1}}{\partial x \partial y}(1-\alpha_i) + (1-A_{i-1})\frac{\partial^2 \alpha_i}{\partial x \partial y} - \frac{\partial A_{i-1}}{\partial x}\frac{\partial \alpha_i}{\partial y} - \frac{\partial A_{i-1}}{\partial y}\frac{\partial \alpha_i}{\partial x} \tag{32}$$

With this formulation, the computation of the image gradients $\frac{\partial I(x,y)}{\partial x}$, $\frac{\partial I(x,y)}{\partial y}$, $\frac{\partial^2 I(x,y)}{\partial x \partial y}$ can be integrated into the rendering loop of 3DGS without much overhead.

## B. Backward Pass

For the backward pass, the partial derivatives of the image gradients with respect to the 2D Gaussian parameters must be computed. As in 3DGS, we use back-to-front blending in the backward pass:

$$I(x,y) = \hat{B}_1 \tag{33}$$

$$\hat{B}_i = (1-\alpha_i)\hat{B}_{i+1} + \alpha_i c_i \tag{34}$$

$$\hat{B}_{N+1} = 0 \tag{35}$$

The derivatives with respect to the screen space positions are

$$\frac{\partial I(x,y)}{\partial x} = \frac{\partial \hat{B}_1}{\partial x} \tag{36}$$

$$\frac{\partial \hat{B}_i}{\partial x} = (1-\alpha_i)\frac{\partial \hat{B}_{i+1}}{\partial x} + \frac{\partial \alpha_i}{\partial x}(c_i - \hat{B}_{i+1}) \tag{37}$$

$$\frac{\partial \hat{B}_i}{\partial y} = (1-\alpha_i)\frac{\partial \hat{B}_{i+1}}{\partial y} + \frac{\partial \alpha_i}{\partial x}(c_i - \hat{B}_{i+1}) \tag{38}$$

$$\frac{\partial^2 \hat{B}_i}{\partial x \partial y} = (1-\alpha_i)\frac{\partial^2 \hat{B}_{i+1}}{\partial x \partial y} + \frac{\partial^2 \alpha_i}{\partial x \partial y}(c_i - \hat{B}_{i+1}) - \frac{\partial \alpha_i}{\partial y}\frac{\partial \hat{B}_{i+1}}{\partial x} - \frac{\partial \alpha_i}{\partial x}\frac{\partial \hat{B}_{i+1}}{\partial y} \tag{39}$$

The gradients for $\sigma, \Sigma, \mu_x, \mu_y$ are computed analogously. We demonstrate the computation of $\sigma$ as an example. The chain rule is applied to compute the partial derivatives of $\sigma_k$:

$$\frac{\partial I(x,y)}{\partial \sigma_k} = \frac{\partial I(x,y)}{\partial \alpha_k}\frac{\partial \alpha_k}{\partial \sigma_k} \tag{40}$$

$$\frac{\partial^2 I(x,y)}{\partial \sigma_k \partial x} = \frac{\partial^2 I(x,y)}{\partial \alpha_k \partial x}\frac{\partial \alpha_k}{\partial \sigma_k} + \frac{\partial I(x,y)}{\partial \alpha_k}\frac{\partial^2 \alpha_k}{\partial \sigma_k \partial x} \tag{41}$$

$$\frac{\partial^2 I(x,y)}{\partial \sigma_k \partial y} = \frac{\partial^2 I(x,y)}{\partial \alpha_k \partial y}\frac{\partial \alpha_k}{\partial \sigma_k} + \frac{\partial I(x,y)}{\partial \alpha_k}\frac{\partial^2 \alpha_k}{\partial \sigma_k \partial y} \tag{42}$$

$$\frac{\partial^3 I(x,y)}{\partial \sigma_k \partial y \partial x} = \frac{\partial^3 I(x,y)}{\partial \alpha_k \partial y \partial x}\frac{\partial \alpha_k}{\partial \sigma_k} + \frac{\partial^2 I(x,y)}{\partial \alpha_k \partial y}\frac{\partial^2 \alpha_k}{\partial \sigma_k \partial x} + \frac{\partial^2 I(x,y)}{\partial \alpha_k \partial x}\frac{\partial^2 \alpha_k}{\partial \sigma_k \partial x} + \frac{\partial I(x,y)}{\partial \alpha_k}\frac{\partial^3 \alpha_k}{\partial \sigma_k \partial x \partial y} \tag{43}$$

The derivative of $I(x,y)$ with respect to $\alpha_a$ is given by

$$\frac{\partial I(x,y)}{\partial \alpha_k} = \frac{\partial \hat{B}_1}{\partial \alpha_k} \tag{44}$$

$$\frac{\partial \hat{B}_i}{\partial \alpha_k} = \delta_{i<k}\left(\frac{\partial \hat{B}_{i+1}}{\partial \alpha_k}(1-\alpha_i)\right) + \delta_{i=k}(c_i - \hat{B}_{i+1}) \tag{45}$$

$$= \prod_{j=1}^{k-1}(1-\alpha_j)(c_k - \hat{B}_{k+1}) \tag{46}$$

$$= (1-A_k)(c_k - \hat{B}_{k+1}) \tag{47}$$

Here, $\delta_{i,j}$ is the Kronecker delta, which equals one if the condition in the subscript is true and zero otherwise. Following this, we calculate the derivative with respect to the screen positions $x, y$:

$$\frac{\partial^2 \hat{B}_i}{\partial \alpha_k \partial x} = -\frac{\partial A_k}{\partial x}(c_k - \hat{B}_{k+1}) - (1-A_k)\frac{\partial \hat{B}_{k+1}}{\partial x} \tag{48}$$

$$\frac{\partial^2 \hat{B}_i}{\partial \alpha_k \partial y} = -\frac{\partial A_k}{\partial y}(c_k - \hat{B}_{k+1}) - (1-A_k)\frac{\partial \hat{B}_{k+1}}{\partial y} \tag{49}$$

$$\frac{\partial^3 \hat{B}_i}{\partial \alpha_k \partial x \partial y} = -\frac{\partial^2 A_k}{\partial x \partial y}(c_k - \hat{B}_{k+1}) - (1-A_k)\frac{\partial^2 \hat{B}_{k+1}}{\partial x \partial y} + \frac{\partial A_k}{\partial x}\frac{\partial \hat{B}_{k+1}}{\partial y} + \frac{\partial A_k}{\partial y}\frac{\partial \hat{B}_{k+1}}{\partial x} \tag{50}$$

**Splat Color Gradients** The calculation of the partial derivatives of a Gaussian's color is straightforward, i.e.,

$$\frac{\partial \hat{B}_i}{\partial c_k} = \delta_{i<k} \frac{\partial \hat{B}_{i+1}}{\partial \alpha_k}(1-\alpha_i) + \delta_{i=k}\alpha_i \tag{51}$$

$$= \prod_{j=1}^{k-1}(1-\alpha_j)\alpha_k = (1-A_k)\alpha_k \tag{52}$$

Differentiating with respect to the screen space position yields the following partial derivatives:

$$\frac{\partial \hat{B}_i}{\partial c_k \partial x} = (1-A_k)\frac{\partial \alpha_k}{\partial x} - \frac{\partial A_k}{\partial x}\alpha_k \tag{53}$$

$$\frac{\partial \hat{B}_i}{\partial c_k \partial y} = (1-A_k)\frac{\partial \alpha_k}{\partial y} - \frac{\partial A_k}{\partial y}\alpha_k \tag{54}$$

$$\frac{\partial^3 \hat{B}_i}{\partial c_k \partial x \partial y} = (1-A_k)\frac{\partial^2 \alpha_k}{\partial x \partial y} - \frac{\partial^2 A_k}{\partial x \partial y}\alpha_k - \frac{\partial A_k}{\partial y}\frac{\partial \alpha_k}{\partial x} - \frac{\partial A_k}{\partial x}\frac{\partial \alpha_k}{\partial y} \tag{55}$$

**Inversion Trick** In the forward pass, we store $A_N, \frac{\partial A_N}{\partial x}, \frac{\partial A_N}{\partial y}, \frac{\partial^2 A_N}{\partial x \partial y}$ for each pixel. To calculate $A_i$ and its derivatives in the backward pass we use the Inversion trick [8]:

$$A_{i-1} = \frac{A_i \alpha_i}{1-\alpha_i} \tag{56}$$

$$\frac{\partial A_{i-1}}{\partial x} = \frac{1}{1-\alpha_i}\left(\frac{\partial A_i}{\partial x} - (1-A_{i-1})\frac{\partial \alpha_i}{\partial x}\right) \tag{57}$$

$$\frac{\partial A_{i-1}}{\partial y} = \frac{1}{1-\alpha_i}\left(\frac{\partial A_i}{\partial y} - (1-A_{i-1})\frac{\partial \alpha_i}{\partial y}\right) \tag{58}$$

$$\frac{\partial A_{i-1}}{\partial x \partial y} = \frac{1}{1-\alpha_i}\left(\frac{\partial^2 A_i}{\partial x \partial y} - (1-A_{i-1})\frac{\partial^2 \alpha_i}{\partial x \partial y} + \frac{\partial A_{i-1}}{\partial x}\frac{\partial \alpha_i}{\partial y} - \frac{\partial A_{i-1}}{\partial y}\frac{\partial \alpha_i}{\partial x}\right) \tag{59}$$

## 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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 2022. IEEE. 3, 4

[2] Pascal Getreuer. Linear Methods for Image Interpolation. *Image Processing On Line*, 1:238–259, 2011. https://doi.org/10.5201/ipol.2011.g_lmii. 2

[3] Gabriel Gouvine. NinaSR: Efficient small and large convnets for super-resolution. https://github.com/Coloquinte/torchSR/blob/main/doc/NinaSR.md, 2021. 3

[4] Gabriel Gouvine. Super-resolution networks for PyTorch. https://github.com/Coloquinte/torchSR, 2021. 3

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4), 2023. Place: New York, NY, USA Publisher: Association for Computing Machinery. 3

[6] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017. 3

[7] William S Russell. Polynomial interpolation schemes for internal derivative distributions on structured grids. *Applied Numerical Mathematics*, 17(2):129–171, 1995. 7

[8] Sebastian Weiss and Rüdiger Westermann. Differentiable Direct Volume Rendering. In *IEEE Transactions on Visualization and Computer Graphics*, pages 562–572, 2022. Issue: 1. 9

[9] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2024. 2, 3

[10] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross. EWA volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, San Diego, CA, USA, 2001. IEEE. 7