Stochastic Ray Tracing for the Reconstruction of 3D Gaussian Splatting

Supplementary Material

A. Proof of Unbiasedness

We now provide a proof of the unbiasedness for our Monte Carlo estimators $\langle \partial_{\boldsymbol{c}} C \rangle$ and $\langle \partial_{\boldsymbol{\alpha}} C \rangle$ presented in Section 4.1 of the main paper. We recall that these estimators work by first drawing an index I and then setting the I-th components using Eq. (8) and Eq. (11) of the main paper while leaving all the other components to zero.

Then, it holds that

$$\mathbb{E}[\langle \partial_{\boldsymbol{c}} C \rangle_i] = \mathbb{P}[I = i] \cdot 1, \tag{1}$$

and

$$\begin{aligned}\mathbb{E}[\langle \partial_{\boldsymbol{\alpha}} C \rangle_i] &= \mathbb{P}[I = i] \cdot \mathbb{E}\left[\frac{c_I - c_K}{\alpha_I} \,\middle|\, I = i \right] \\ &= \mathbb{P}[I = i] \, \frac{c_i - \mathbb{E}[c_K \,|\, I = i]}{\alpha_i} \\ &= \frac{\mathbb{P}[I = i]}{\alpha_i} \left(c_i - \sum_{k \succ i} \mathbb{P}[K = k \,|\, I = i] \, c_k \right).\end{aligned} \tag{2}$$

for all $i = 1, 2, \ldots, n$.

We recall that, according to Eq. (4) and Eq. (10) of the main paper, we have

$$\mathbb{P}[I = i] = \alpha_i \prod_{j \prec i} (1 - \alpha_j), \tag{3}$$

$$\mathbb{P}[K = k \,|\, I = i] = \alpha_k \prod_{i \prec t \prec k} (1 - \alpha_t). \tag{4}$$

Substituting Eq. (3) and Eq. (4) into Eq. (1) and Eq. (2) yields

$$\mathbb{E}[\langle \partial_{\boldsymbol{c}} C \rangle_i] = \alpha_i \prod_{j \prec i} (1 - \alpha_j), \tag{5}$$

and

$$\begin{aligned}&\mathbb{E}[\langle \partial_{\boldsymbol{\alpha}} C \rangle_i] = \\ &\left(\prod_{j \prec i} (1 - \alpha_j) \right) \left(c_i - \sum_{k \succ i} c_k \alpha_k \prod_{i \prec t \prec k} (1 - \alpha_t) \right).\end{aligned} \tag{6}$$

Eq. (5) and Eq. (6) match the desired partial derivatives ${}^{\partial C}/_{\partial c_i}$ and ${}^{\partial C}/_{\partial \alpha_i}$ given by Eq. (6) and Eq. (7) of the main paper, respectively.

B. Comparison to StochasticSplats

Proposed by Kheradmand et al. [1], StochasticSplats introduces a stochastic method for estimating gradients of the alpha-blended pixel colors C with respect to the Gaussian parameters. In what follows, we present the pseudo-code for the alternative estimator in Algorithm 1, and compare the efficiency of both estimators via empirical experiments.

Algorithm 1: Monte Carlo estimator for pixel color gradient $\partial_{\boldsymbol{\alpha}} C$ introduced by StochasticSplats [1].

$\langle \partial_{\boldsymbol{\alpha}} C \rangle^{\text{ssplats}} \leftarrow \mathbf{0}$;
for $m = 1$ *to* M_b **do**
  /* Draw index I */
  $I \leftarrow 0$; $z \leftarrow \infty$; $\alpha \leftarrow 1$;
  foreach *Gaussian* g_i *along the camera ray* **do**
    Obtain its opacity α_i and depth z_i;
    Draw ξ uniformly from $[0, 1)$;
    if $\xi < \alpha_i$ *and* $z_i < z$ **then**
      $I \leftarrow i$; $z \leftarrow z_i$; $\alpha \leftarrow \alpha_i$;
  if $I > 0$ **then**
    /* Obtain Gaussian color */
    Compute the color c_I of the Gaussian g_I;
    foreach *Gaussian* g_k *along the camera ray* **do**
      Obtain its opacity α_k and depth z_k;
      if $z_k < z_I$ **then**
        /* Update the gradient estimates */
        $\langle \partial_{\boldsymbol{\alpha}} C \rangle_k^{\text{ssplats}} \mathrel{+}= {}^{-c_I}/_{(1-\alpha_k)}$; // Eq. (8)
    /* Update the gradient estimates */
    $\langle \partial_{\boldsymbol{\alpha}} C \rangle_I^{\text{ssplats}} \mathrel{+}= {}^{c_I}/_{\alpha}$; // Eq. (7)
return ${}^{\langle \partial_{\boldsymbol{\alpha}} C \rangle^{\text{ssplats}}}/_{M_\text{b}}$;

B.1. Alternative Gradient Estimator

StochasticSplats [1] uses the same procedure as our method to estimate the gradient $\partial_{\boldsymbol{c}} C$ but a different one for $\partial_{\boldsymbol{\alpha}} C$. In the following, we focus on the estimation of the latter.

As described in Algorithm 1, to estimate the gradient $\partial_{\boldsymbol{\alpha}} C := ({}^{\partial C}/_{\partial \alpha_1}, {}^{\partial C}/_{\partial \alpha_2}, \ldots, {}^{\partial C}/_{\partial \alpha_n})$ with respect to the Gaussian opacities $\alpha_1, \alpha_2, \ldots, \alpha_n$, StochasticSplats starts with drawing an index I the same way as our method, and setting

$$\langle \partial_{\boldsymbol{\alpha}} C \rangle_I^{\text{ssplats}} = \frac{c_I}{\alpha_I}. \tag{7}$$

Then, instead of randomly drawing some $K \succ I$, StochasticSplats lets

$$\langle \partial_{\boldsymbol{\alpha}} C \rangle_k^{\text{ssplats}} = \frac{-c_I}{1 - \alpha_k}, \tag{8}$$

for all Gaussians g_k located in front of the Gaussian g_I (i.e., with $k \prec I$).

Table 1. **Novel view synthesis**: Comparison between our method, our method with stochastic forward pass, and [1], in terms of both speed and quality.

Method\Metric	MipNeRF360				Tanks & Temples				Deep Blending			
	PSNR↑	SSIM↑	LPIPS↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Time↓
Ours	28.31	0.857	0.254	33m	22.57	0.830	0.221	20m	29.87	0.906	0.324	25m
Ours-FullStoch (15spp)	27.90	0.843	0.265	37m	22.37	0.821	0.233	21m	29.51	0.902	0.330	24m
Ours-Fullstoch (30spp)	28.14	0.850	0.254	52m	22.75	0.827	0.229	25m	29.75	0.904	0.327	29m
SS-Tracing	22.12	0.648	0.451	89m	14.94	0.532	0.560	54m	18.08	0.726	0.578	68m

Although this approach returns denser gradient estimates by spatting gradients to not only the I-th component but also all $k \prec I$, it suffers from two major drawbacks. First, when executed on the GPU, writing to all $k \prec I$ components in parallel requires significantly more atomic operations, which can lead to reduced computational efficiency. Second, the division by $(1-\alpha_k)$ in Eq. (8) can produce near-infinite gradients—and therefore high variance—whenever high-opacity Gaussians make $\alpha_k \approx 1$, which occurs frequently.

The key difference between our method and [1] is the order of derivation. StochasticSplats [1] first designs a Monte Carlo estimator for alpha blending, and subsequently takes the derivative of the estimator. Our method, in contrast, first takes the derivative of alpha blending, and designs a Monte Carlo estimator for the derivative. This ordering introduces a term $1/1-\alpha_{K \prec I}$ in the density gradient, where the denominator easily evaluates to a near-0 value when an opaque Gaussian exists, causing problematic gradients.

B.2. Comparisons

To compare the efficiency of our gradient estimator (Algorithm 2 of the main paper) and the alternative one proposed by StochasticSplats (Algorithm 1), we implement the latter on our stochastic ray tracing codebase, which we refer to as *SS-Tracing*.

We give an intuition on the variance of both methods in Figure 1 by visualizing the gradient image. The images show the magnitude of screen space gradient with respect to the horizontal motion of the Gaussians, as well as a plot of the opacities of Gaussians along a ray. Our experiment shows that high opacity Gaussians frequently occur, and can easily lead to large variance for *SS-Tracing*. Empirically, it takes $16\times$ samples for SS-Tracing to reach the same level of variance as our method.

We compare the numerical metrics of our method and our re-implemented *SS-Tracing* in Table 1.

C. Additional Results

We provide additional evaluation of our method, following Section 5 of the main text.

In Table 1, we provide an ablation study of an important variant of our algorithm. We replace the sorting-based forward pass with the stochastic algorithm in Algorithm 1 of the main paper, proposed by Sun et al. [2], vary the number of samples used in the forward pass and evaluate the performance on novel view synthesis tasks. Experiment shows that our hybrid approach consistently outperforms the full stochastic variants in terms of both reconstruction quality and time.

Table 2. NeRF Synthetic Dataset

	PSNR↑	**SSIM↑**	**LPIPS↓**	**Time↓**
3DGRT	33.84	0.970	0.038	10m21s
SS-Tracing	31.67	0.958	0.050	24m31s
Ours	33.99	0.970	0.039	6m57s

In Table 2, we show the metrics of our method, 3DGRT and SS-Tracing on the NeRF-Synthetic dataset. Our performance is consistent with the results we show in the main paper.

In Figure 2, we demonstrate that our method is seamlessly compatible with distorted camera models due to the usage of ray tracing.

In Figure 3, we show additional results of the equal-time novel view synthesis benchmark. In Figure 4, we show additional results from relightable Gaussian splatting evaluations.

References

[1] Shakiba Kheradmand, Delio Vicini, George Kopanas, Dmitry Lagun, Kwang Moo Yi, Mark Matthews, and Andrea Tagliasacchi. Stochasticsplats: Stochastic rasterization for sorting-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 1, 2, 3

[2] Xin Sun, Iliyan Georgiev, Yun (Raymond) Fei, and Miloš Hašan. Stochastic Ray Tracing of Transparent 3D Gaussians. In *Eurographics Symposium on Rendering*. The Eurographics Association, 2025. 2

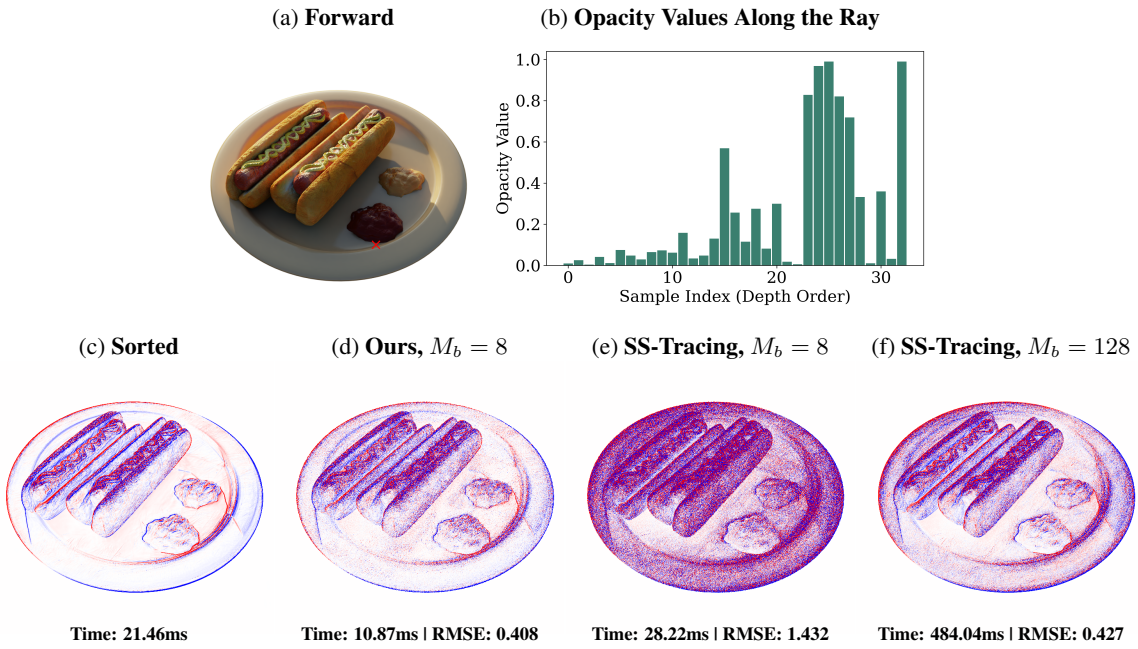

Figure 1. Visualization of the variance of the gradient obtained by our method and the baseline method. (a) shows the forward rendering result, and (b) visualizes the opacities of each Gaussian along the ray at the red cross in (a). We further visualize the gradient that the Gaussians along each pixel receive when applying: (c) Sorted alpha blending (3DGRT), (d) Ours, with $M_b = 8$, (e) Our implementation of [1], with $M_b = 8$, and (f) Our implementation of [1], with $M_b = 128$. The time for the backward pass and RMSE are provided with each image.

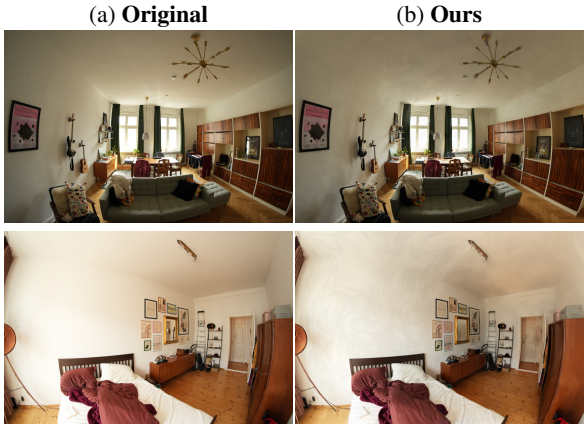

Figure 2. Similar to our ray tracing baseline, our method also seamlessly supports distorted cameras, e.g., fisheye.

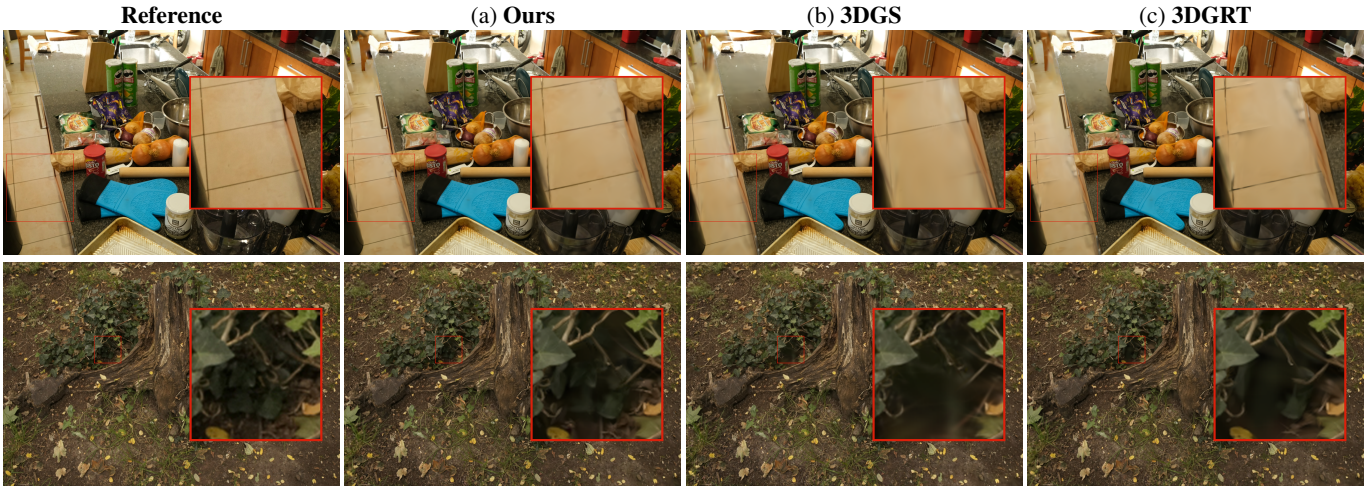

Figure 3. **Equal-time Comparison** between our method and baselines. We run the methods for the same period of wall-clock time, and compare the reconstruction quality. Our method produces comparable visual quality to 3DGS, and outperforms 3DGRT.

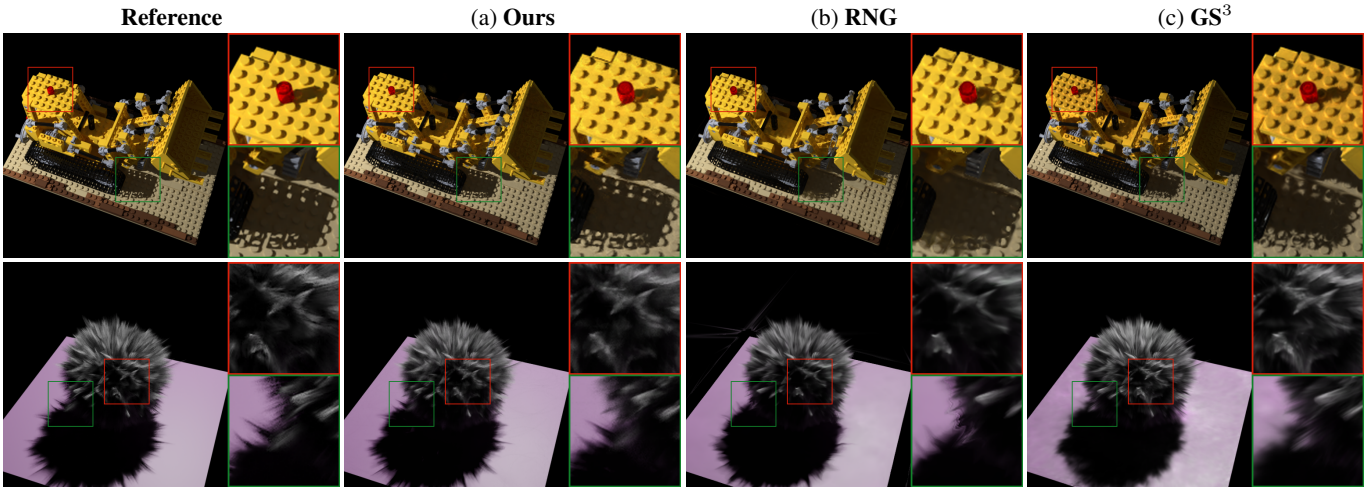

Figure 4. Visualization of reconstruction quality of our method and baselines. Our method produces significantly better geometry, and in particular, shadow quality.