

Depth Peeling for High-Fidelity Gaussian-Enhanced Surfel Rendering

Supplementary Material

In the supplementary material, Sec. 8–Sec. 11 describe more technical details, while the remaining sections present additional experimental results.

8. Determining Depth Margin

In Sec. 4, we use the depth margin ϵ_s to prevent Gaussians intersecting with surfels from being partially truncated during the depth test. Specifically, we first compute the initial value of $\epsilon_i = \frac{5}{2}(s_i^X + s_i^Y)$ in the same way as GES, and then, for each surfel, we query its 16 nearest neighboring surfels and take the median of their ϵ values as the final one. Our motivation stems from the fact that using a fixed constant margin can prevent truncation but may lead to occlusion leakage in fine structures, as discussed in GES [42]. To address this, GES defines the margin based on surfel scaling, achieving geometry-adaptive control to mitigate such leakage. However, we observed that this strategy is not sufficiently robust: when a large surfel is located near fine structures composed of many small surfels, its large depth margin can still cause occlusion leakage. Our modification enforces a locally smooth distribution of ϵ , further reducing the probability of such leakage.

9. Training Details

Joint Optimization. During this stage, we adopt the densification and pruning strategies of GES with minor modifications. Specifically, for surfels, we prune surfels that are either heavily occluded or extremely small, determined by the number of pixels they cover in the first peeled layer. The pruning thresholds are the same as GES.

For Gaussians, we introduce an extra step to split overly large Gaussians. We found that DBS [25] tends to generate many excessively large Gaussians due to the added positional noise, and such Gaussians often intersect with surfels, causing partial occlusion artifacts (see Fig. 13). Although our depth tolerance margin ϵ helps mitigate this issue, it remains ineffective for extremely large Gaussians. Therefore, every 2000 iterations, we query the 16 nearest surfels for each Gaussian, compute the average ϵ among them, and if a Gaussian’s average scale exceeds twice this value, we identify it as a potentially oversized Gaussian and split it into two smaller Gaussians following the same splitting strategy as in 3DGS [19]. For the depth margin ϵ , since the surfel scale remains stable after initialization, it is computed only at the beginning and once again at the 4k-th iteration of joint optimization. The joint optimization converges after around 25k iterations on average.

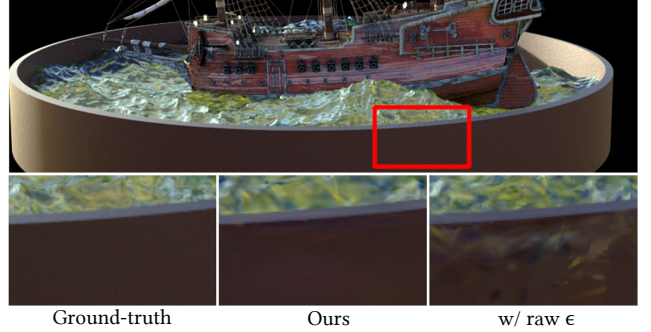


Figure 12. Qualitative comparisons of images rendered with the original ϵ strategy used in GES [42].

Initialization. Although our surfels are differentiable within their semi-transparent boundary regions, their optimization efficiency remains limited compared to fully continuous representations. Therefore, similar to GES, we initialize the surfel positions and scales using 2DGS [17]. Specifically, we take the sparse SfM point cloud as the initial input and train a 2DGS model for 20k iterations using the same differentiable rendering pipeline and loss functions as in the original 2DGS. The positions and scales of the final 2D Gaussians with opacity values greater than 0.8 are then used to initialize our surfels. This threshold is the same as in GES [42].

For Gaussians, we incorporate the MCMC-based densification from DBS [25] via a fast initialization step to control the primitive count and enhance reconstruction quality. Specifically, using the sparse SfM point cloud as input, we train DBS for 20k iterations with its Beta kernel replaced by a fixed Gaussian kernel, utilizing the resulting positions and scales for our Gaussian initialization. To accelerate this process, the initial number of Gaussians is kept significantly smaller than the final target. Following this MCMC phase, we continue adding Gaussians based on image-space errors during training, identically to GES, until the predefined limit is reached.

10. OpenGL Implementation

As demonstrated in Sec. 5, we develop an OpenGL-based renderer for real-time rendering after training, fully leveraging the efficiency of our sort-free rendering scheme within the traditional graphics pipeline. The OpenGL-based renderer also consists of two main passes:

In the first pass, we perform depth peeling for the nearest three surfel layers. During each rasterization step, we write not only the depth but also the surfel ID into the buffer. This



Figure 13. Qualitative comparisons of images rendered without our large Gaussian splitting strategy during training.

additional ID information enables us to correctly identify and handle interleaved surfels, preventing the issue where two intersecting surfels could otherwise be peeled only once along their intersection line. After the three peeling passes, a post-processing shader blends the peeled layers to generate the final surfel color map, while also writing the ϵ -adjusted depth into the depth buffer. The peeled depth maps $\{D_i^s\}_{i=1}^2$ and transmittance maps $\{T_i^s\}_{i=1}^2$ are packed into a four-channel texture. Since Gaussians beyond D_3^s are culled by the hardware depth test, storing D_3^s and T_3^s is unnecessary.

In the second pass, Gaussians are splatted onto the screen with depth testing enabled and depth writing disabled. During this process, the shader queries the peeled depth and transmittance values from the four-channel texture to accumulate color and weight contributions. Finally, another post-processing shader performs weight normalization, producing the final rendering result. Our method is entirely based on the programmable graphics pipeline and does not rely on compute shaders, making it easy to be integrated into existing rendering engines and mobile devices.

11. Geometry Regularization

To enable the surfel representation to better capture the scene geometry, which is essential for downstream tasks such as mesh extraction, we optionally adopt a geometry regularization loss $\mathcal{L}_{geo} = \lambda_4 \mathcal{L}_{sd} + \lambda_5 \mathcal{L}_{sn} + \lambda_6 \mathcal{L}_{sdn}$, where $\lambda_4 = 0.1$, $\lambda_5 = 0.05$ and $\lambda_6 = 0.05$, respectively.

$$\mathcal{L}_{sd} = \mathcal{L}_1(D_{supv}, D_s), \quad (6)$$

$$\mathcal{L}_{sn} = \frac{1}{HW} \sum_{\hat{\mathbf{x}}} (1 - N_{supv}(\hat{\mathbf{x}}) \cdot N_s(\hat{\mathbf{x}})), \quad (7)$$

$$\mathcal{L}_{sdn} = \frac{1}{HW} \sum_{\hat{\mathbf{x}}} (1 - N_{supv}(\hat{\mathbf{x}}) \cdot \nabla(D_s(\hat{\mathbf{x}}))), \quad (8)$$

where $\nabla(\cdot)$ is the depth-to-normal operator, the reference depth maps D_{supv} and normal maps N_{supv} are generated in

Table 4. Ablation study. Dataset-averaged image quality and frame rate (1080p) on Mip-NeRF360 Dataset [2]. Base/Full: using our base/full model; w/o splitting: without the large-Gaussian splitting strategy; w/ raw ϵ : using the original ϵ computation from GES [42].

	SSIM \uparrow	PSNR \uparrow	LPIPS \downarrow	FPS \uparrow
GES	0.813	27.38	0.208	675
Base, w/o trans. grad	0.818	27.57	0.201	482
Base	0.821	27.72	0.197	480
Base, w/ MCMC dens.	0.826	27.98	0.195	470
Full	0.827	28.11	0.196	472
Full, w/o splitting	0.825	28.04	0.197	479
Full, w/ raw ϵ	0.827	28.08	0.199	469

the last iteration of 2DGS initialization, and the depth map D_s and normal map N_s of surfels are rendered by alpha blending, following the same procedure as Eq. (1).

12. Additional Comparisons with GES

Surfel-Gaussian Decomposition Rendering. In the leftmost 3 columns of Fig. 14, we visualize the surfel-Gaussian decomposition rendering compared with GES. In *Chair*, GES preserves high-frequency texture in surfel colors because its image-loss-based surfel initialization inevitably embeds appearance details, deviating from the intended coarse-geometry representation. In contrast, DP-GES naturally transfers high-frequency details from surfels to Gaussians during joint training, resulting in smoother surfel rendering without visual noise (see the zoomed-in patches of *Hotdog* and *Drums*).

Surfel Distribution and Geometry Reconstruction. In this experiment, we compare our DP-GES with GES, with geometry regularization enabled for both methods. In the rightmost 3 columns of Fig. 14, the joint optimization in DP-GES keeps surfels well-regularized, yielding a more uniform and coherent distribution than GES. This prevents geometry and appearance artifacts: GES often produces needle-like surfels that manifest as sharp spikes, resulting in non-smooth surfaces in the extracted mesh.

13. Additional Ablations

We present quantitative additive ablations and additional results in Table 4. The additive study demonstrates a clear and continuous improvement as we progressively upgrade the original GES to our full model. Applying our rendering model to GES while disabling the new gradient path (*Base, w/o trans. grad*) alleviates the aliasing issue, yet the reconstruction remains suboptimal. Enabling the gradient path (*Base*) enhances the reconstruction quality. Incorporating the MCMC-based densification strategy from DBS [25]

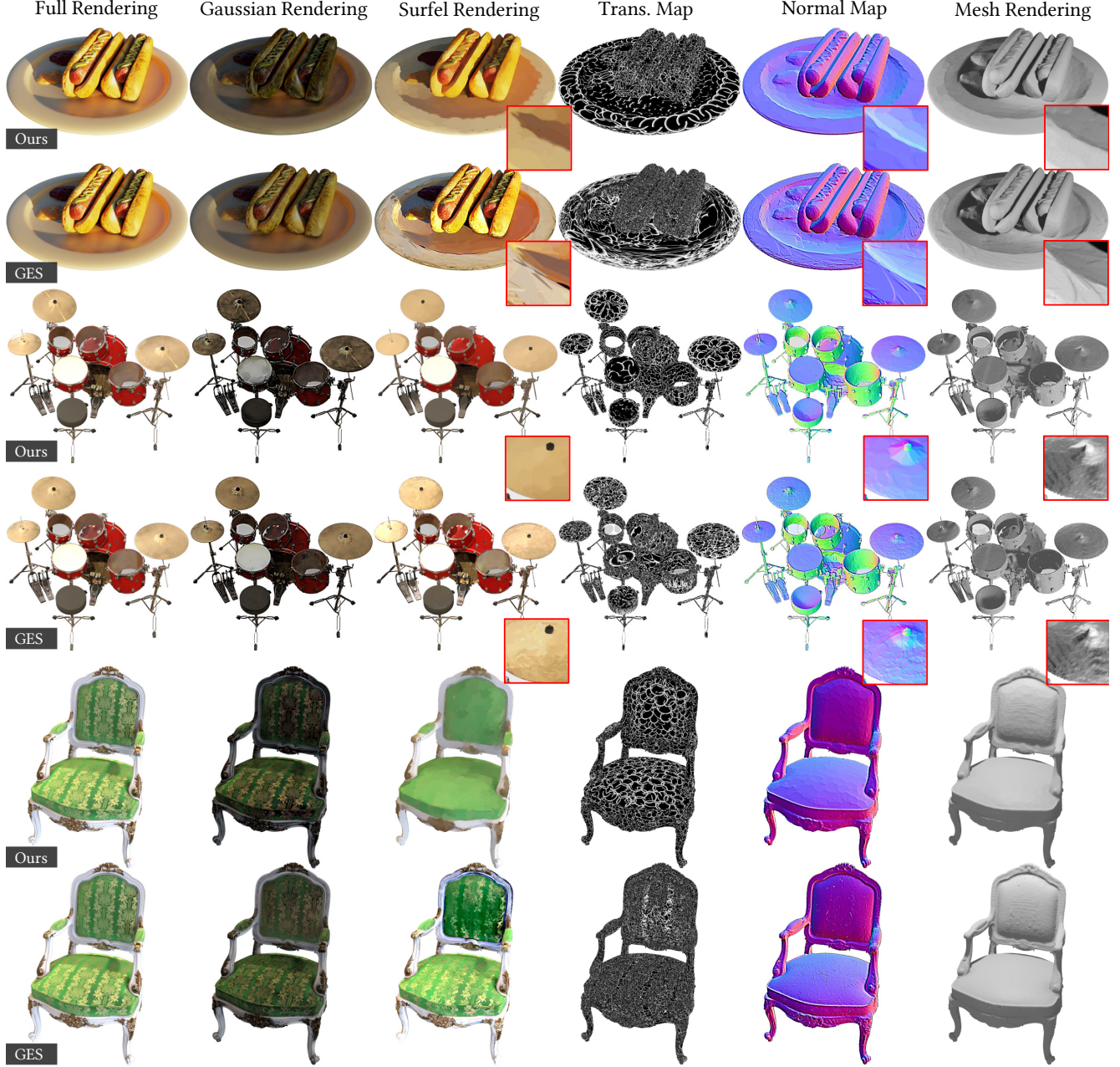


Figure 14. Qualitative comparisons between our DP-GES and GES [42] are shown across three aspects: surfel-Gaussian decomposition rendering, surfel distribution, and surfel geometry. For decomposition rendering, we visualize the Gaussian and surfel components as $C_G/(W_s + W_G)$ and $C_s/(W_s + W_G)$, respectively. Surfel distribution is illustrated by the transmittance map T_1^s in DP-GES, with GES processed in a similar way. Surfel geometry is represented by the normal map computed from depth and the mesh extracted from surfels. From top to bottom: scenes are *Hotdog*, *Drums* and *Chair* from the NeRF Synthetic Dataset [29].

(*Base*, *w/o MCMC dens.*) yields additional gains, and replacing the SH color with SB color representation (*Full*) brings a marginal but consistent quality increase. These steady improvements validate that our core contributions are highly effective and orthogonal to existing auxiliary techniques.

Impact of Improved Depth Margin Strategy. In Fig. 12, smooth structures (*e.g.*, pot rims) and detailed structures (*e.g.*, waves) are typically fitted by large and small surfels, respectively. Using the same ϵ strategy as in GES, occlusion leakage occurs in regions where large and small surfels are adjacent. Our ϵ computation incorporates the sizes

of neighboring surfels, yielding smoother local transitions and reducing such leakage.

Impact of Large Gaussian Splitting. In Fig. 13, our Gaussian splitting strategy mitigates the issue of excessively large Gaussians produced during DBS initialization. Without splitting, these large Gaussians can extend across surfels and be abruptly truncated by the depth test, producing visual artifacts. By splitting them into smaller Gaussians, we reduce the probability of such depth-test cutoffs.