Gaussians Can Subtract: Neural Dissolve Fields for Hybrid Scene Rendering

Supplementary Material

S1. Additional Method Details

S1.1. Fixed Mesh Depth in Continuous Fusion

A possible extension of dissolution is to recompute a new depth map after the mesh has been partially dissolved, and then use this updated depth to separate front and back Gaussian contributions. We do not adopt this design. Once dissolution becomes soft, a unique post-dissolution depth is no longer naturally defined. Any such depth would require an additional heuristic, such as thresholding a soft visibility profile or defining an expected depth along the ray. This would introduce extra hyperparameters and make the front/back partition itself depend on the dissolve prediction.

More importantly, such a design creates an undesirable feedback loop. If dissolution changes the depth, then the updated depth changes which Gaussians are assigned to the front or back segment. Those changed Gaussian contributions in turn modify the gradients that supervise the dissolve field. As a result, the partition boundary itself becomes unstable near occlusion transitions.

For this reason, Gaussolve always uses the rasterized mesh depth $Z_m(\mathbf{u})$ as a fixed depth for the Gaussian-center front/back partition. This depth is used only to compute the front-segment statistics (C_f, T_f) required by the decomposition in the main paper. The dissolve field does not alter $Z_m(\mathbf{u})$. Instead, it only modulates the effective mesh opacity

$$\alpha_m^{\text{eff}}(\mathbf{u}) = 1 - D(\mathbf{u}). \tag{S1}$$

Thus the mesh depth remains fixed for the front/back decomposition, while dissolution only changes the final blending weight.

S1.2. Parallel Rendering Implementation

For each viewpoint, the mesh is rasterized once to obtain the reference depth map $Z_m(\mathbf{u})$, the visible mesh mask, and screen-space coordinates. After this step, the mesh, dissolve, and residual Gaussian branches become largely independent until the final compositing stage. The mesh branch computes the mesh color $C_m(\mathbf{u})$; the dissolve branch renders $D(\mathbf{u})$ and converts it into $\alpha_m^{\text{eff}}(\mathbf{u}) = 1 - D(\mathbf{u})$; and the residual Gaussian branch uses the fixed mesh depth to compute (C_f, T_f) and C_g. The final image is

$$\begin{aligned} I_{\text{hybrid}}(\mathbf{u}) = \alpha_m^{\text{eff}}(\mathbf{u})(C_f(\mathbf{u}) + T_f(\mathbf{u})C_m(\mathbf{u})) \\ + (1 - \alpha_m^{\text{eff}}(\mathbf{u}))C_g(\mathbf{u}). \end{aligned} \tag{S2}$$

After the shared mesh rasterization and depth computation, the three branch computations are scheduled asynchronously on separate CUDA streams.

S1.3. Multi-view Mesh-Texel Initialization

We initialize the NDF around mesh regions that consistently disagree with the training images. Specifically, we use the mesh UV atlas as a common domain for aggregating errors across views by rendering the pretrained textured mesh from every training view and transferring the resulting RGB reconstruction errors to the corresponding mesh texels using the rasterized UV coordinates.

For a texel q, let $\mathcal{V}_q$ denote the training views in which it is visible, and let $e_v(q)$ denote its mean RGB error in view v. Its multi-view error is

$$E(q) = \frac{1}{|\mathcal{V}_q|} \sum_{v \in \mathcal{V}_q} e_v(q). \tag{S3}$$

We select texels with high multi-view error that are observed from multiple training views. For a selected texel q, let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ denote the vertices of its mesh triangle, and let $\beta_1(q), \beta_2(q), \beta_3(q)$ be the corresponding barycentric coordinates. Its 3D surface position is

$$\mathbf{x}(q) = \beta_1(q)\mathbf{x}_1 + \beta_2(q)\mathbf{x}_2 + \beta_3(q)\mathbf{x}_3. \tag{S4}$$

To remove UV-seam duplicates and overly dense samples, each position is assigned to a 3D voxel,

$$\mathbf{k}(q) = \left\lfloor \frac{\mathbf{x}(q) - \mathbf{x}_{\min}}{\delta} \right\rfloor, \tag{S5}$$

where δ is the voxel size and $\mathbf{x}_{\min}$ is the minimum corner of the selected point set. Candidates assigned to the same voxel are merged, retaining the one with the strongest multi-view support. A scalar-SH dissolve Gaussian is initialized at each retained position. These Gaussians are subsequently fully trainable and are not constrained to remain on the mesh surface.

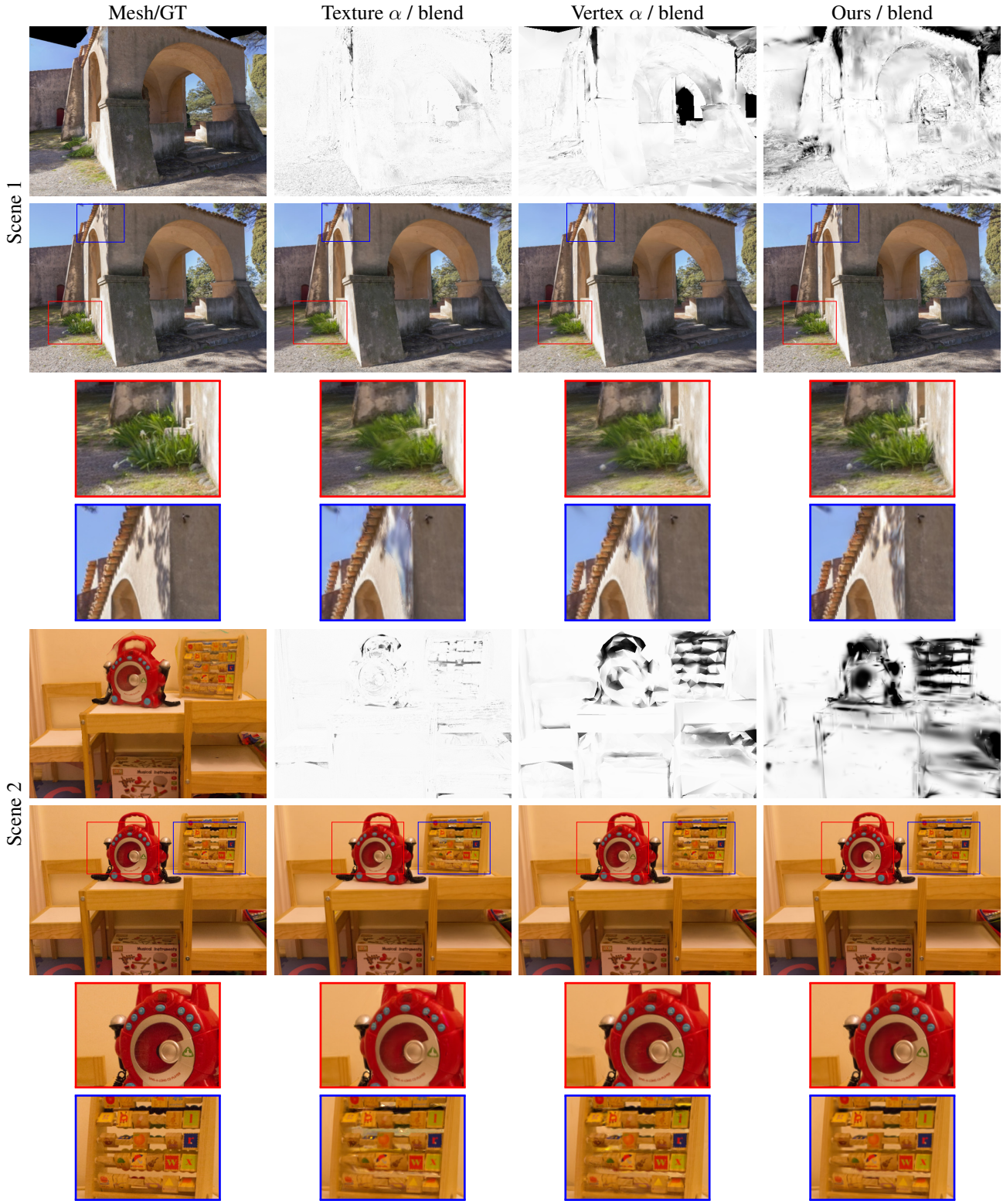


Figure S1. **Surface-alpha dissolve alternatives.** We compare texture-map alpha, per-vertex alpha, and the proposed volumetric NDF. The first row of each scene shows the learned mesh/dissolve representation, the second row shows the blended rendering with red and blue boxes, and the third row shows the corresponding zoom-ins. Texture-map and vertex alpha can reduce some mesh–image conflicts but often produce diffuse or fragmented masks, while the volumetric NDF localizes unreliable mesh evidence and enables cleaner residual Gaussian refill.

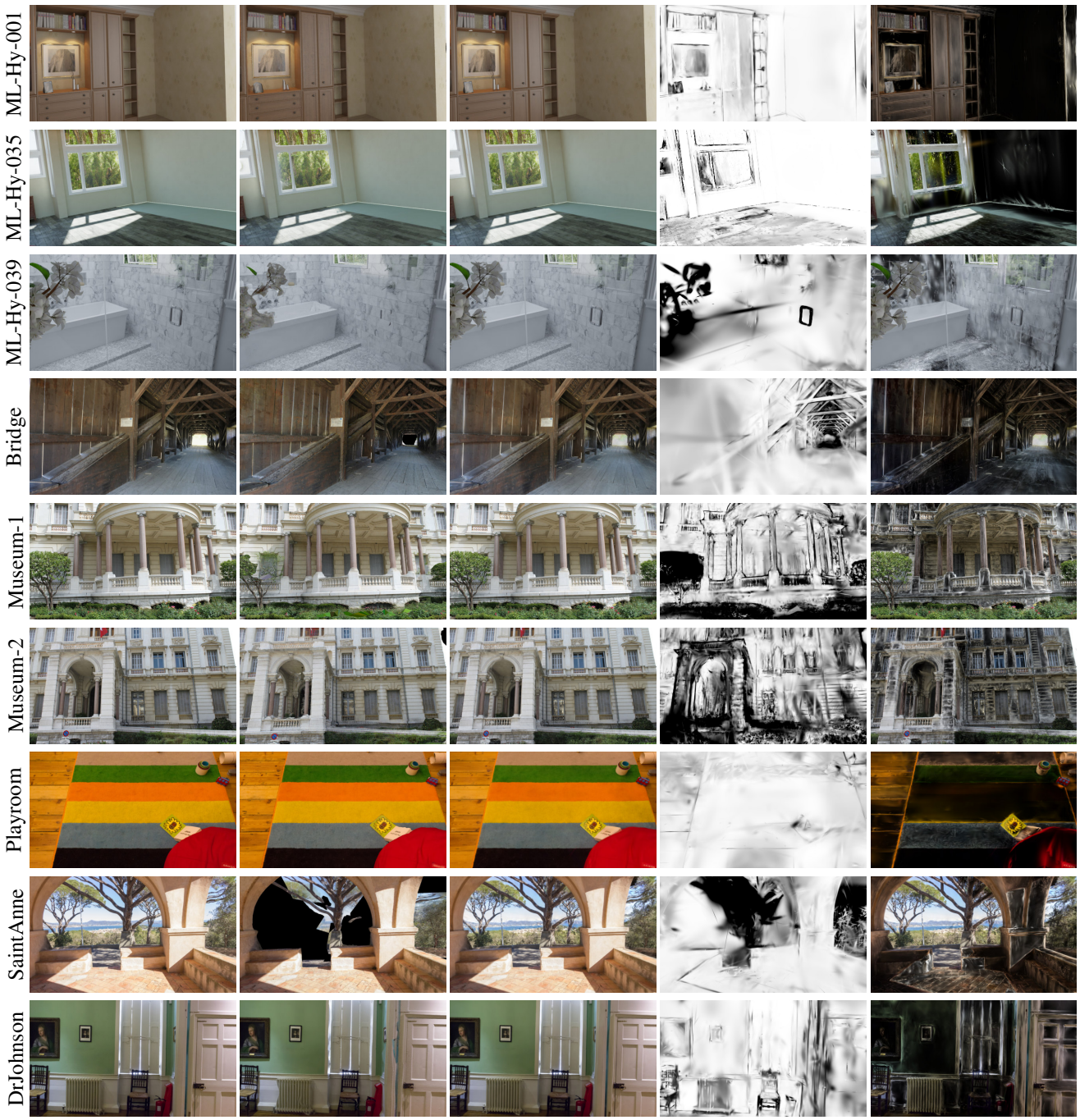


Figure S2. **Per-scene component visualizations.** The decomposition shows how the dissolve field suppresses unreliable mesh evidence and how residual Gaussians refill the corresponding regions. Here, ML-Hy denotes the ML-Hypersim-AI scene family.

S1.4. Loss-Driven Dual Densification

The initial NDF captures mesh–image conflicts that are already visible before hybrid training. However, additional conflicts may emerge as the hybrid representation evolves. We therefore periodically render the current fused result for all training views and compute its per-pixel mean absolute RGB error. Pixels with negligible errors are discarded.

For a training view v, let $R_v(p)$ denote the reconstruction error at pixel p, and let Ω_v denote the set of pixels whose errors exceed the selection threshold. We sample pixels from an error-proportional categorical distribution,



Figure S3. **Additional adjacent-view consistency.** Consecutive rendered crops remain stable across nearby views, including regions where residual Gaussians correct unreliable mesh evidence.

Algorithm S1 Parallel Differentiable Hybrid Rendering

1: **Input:** viewpoint $\mathbf{v}$, mesh prior $\mathcal{M}$, residual Gaussians $\mathcal{G}_r$, dissolve Gaussians $\mathcal{G}_d$
2: Rasterize $\mathcal{M}$ to obtain mesh depth $Z_m(\mathbf{u})$, visibility mask, and screen-space coordinates.
3: Launch three asynchronous branches.
· Mesh branch: render the textured mesh color $C_m(\mathbf{u})$.
· Dissolve branch: rasterize $\mathcal{G}_d$ to obtain $D(\mathbf{u})$ and $\alpha_m^{\text{eff}}(\mathbf{u}) = 1 - D(\mathbf{u})$.
· Residual branch: render $\mathcal{G}_r$ with fixed mesh depth to obtain C_f, T_f, and C_g.
4: Synchronize all streams and composite $I_{\text{hybrid}} = \alpha_m^{\text{eff}}(C_f + T_f C_m) + (1 - \alpha_m^{\text{eff}})C_g$.

$$P_v(p) = \frac{R_v(p)}{\sum_{p' \in \Omega_v} R_v(p')}, \qquad p \in \Omega_v, \tag{S6}$$

Thus, pixels with larger reconstruction errors have a higher probability of being selected.

For each sampled pixel, we use the rasterized mesh depth to obtain a mesh-guided 3D anchor. Let $D_m(p)$ denote the mesh depth at pixel p and $\text{BackProj}_v(p, D_m(p))$ denote its back-projection in view v. The anchor position is

$$\mathbf{x}(p) = \text{BackProj}_v\left(p, D_m(p)\right). \tag{S7}$$

At this position, we initialize one dissolve candidate and one residual RGB candidate. The dissolve candidate increases the capacity to attenuate unreliable mesh evidence, while the residual candidate increases the capacity to reconstruct the content revealed after dissolution.

Candidates collected from different training views are first retained based on visibility across multiple distinct cameras, and then residual candidates are merged using the 3D voxel aggregation described above. Retained residual candidates are initialized from the corresponding ground-truth colors, whereas dissolve candidates start from a small scalar dissolve value. Although both candidates share the same initial mesh-guided anchor, they are optimized independently and are not constrained to remain on the mesh surface.

S2. Additional Implementation Details

S2.1. Training and Reproduction Settings

All locally reproduced methods use the same calibrated cameras, train/test splits, input resolution, and scene coordinate system. Unless otherwise specified, training is performed for 30k iterations on a single NVIDIA RTX 5000 Ada GPU. Test views are not used for mesh reconstruction, texturing, hyperparameter selection, or training.

For each baseline, we retain its official training objective and native Gaussian update strategy. The Gaussolve variants use the same mesh, NDF initialization, continuous fusion, and training schedule; they differ only in the update strategy of the residual Gaussian backbone, as described in Sec. S2.2. Exact commands and configuration files are included in the supplementary code.

The MeshGS and Hybrid Mesh-Gaussian results in the main paper are taken directly from their respective publications and are not included among our local reproductions.

MeshSplatting reproduction. The official MeshSplatting implementation constructs its mesh through a Restricted Delaunay (RDel) stage. Its topology code assumes that each triangular face is shared by at most two tetrahedra. On several large scenes, the raw Delaunay output contained duplicate, degenerate, or non-manifold tetrahedra that violated this assumption and caused the official code to terminate.

For these scenes, we used a deterministic robustness fix before the existing RDel topology construction. We isotropically normalize the input coordinates, canonicalize and deduplicate the generated tetrahedra, remove degenerate tetra-

Table S1. **Quantitative breakdown of hybrid methods on the Deep Blending dataset.** We select four representative scenes encompassing both complex indoor layouts and structured outdoor architectural environments. **Bold** indicates the best performance, and cell colors highlight the top, second-best, and third-best results. † Indicates catastrophic geometry collapse.

Method	Mesh	Hybrid	Indoor: DrJohnson				Indoor: Playroom				Outdoor: SaintAnne				Outdoor: Bridge			
			PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑
3DGS	×	×	29.33	**0.897**	0.271	0%	30.62	**0.897**	0.266	0%	25.54	0.838	0.280	0%	26.44	0.854	0.260	0%
Reduced 3DGS	×	×	29.30	0.894	0.276	53%	30.50	0.896	0.266	61%	25.50	0.835	0.287	39%	25.59	0.837	0.286	59%
3DGS-MCMC	×	×	29.40	0.889	0.274	1%	30.65	0.896	0.252	-36%	25.62	0.840	0.279	0%	**26.46**	**0.856**	0.257	0%
SuGaR	✓	×	28.30	0.870	0.326	59%	30.25	0.884	0.289	37%	24.20	0.820	0.301	-38%	24.50	0.791	0.320	5%
PGSR	✓	×	29.25	0.895	0.273	37%	30.65	0.868	0.318	28%	25.42	0.827	0.315	19%	25.46	0.834	0.291	44%
MeshSplatting	✓	×	27.28	0.837	0.367	–	24.17	0.808	0.365	–	22.69	0.780	0.346	–	22.79	0.748	0.395	–
3D-GES	×	✓	29.42	0.891	0.262	54%	30.66	0.895	0.253	45%	25.18	0.839	0.270	23%	11.75†	0.521†	0.608†	–
Hard-Z Proxy	✓	✓	29.15	0.891	0.244	49%	30.55	0.890	0.226	49%	25.27	0.836	0.231	20%	25.97	0.847	0.241	59%
Ours (3DGS)	✓	✓	29.69	0.894	**0.240**	58%	30.93	0.893	0.223	61%	25.61	0.842	**0.221**	21%	26.23	0.854	**0.234**	58%
Ours (Reduced 3DGS)	✓	✓	29.60	0.895	0.243	**74%**	30.81	0.892	0.229	**76%**	25.62	0.843	0.224	**45%**	26.29	0.850	0.236	**76%**
Ours (3DGS-MCMC)	✓	✓	**29.95**	0.896	0.241	-2%	**31.10**	0.897	**0.218**	-3%	**25.73**	**0.847**	0.222	-5%	26.39	0.850	0.238	-6%

Table S2. **Irregular-scene stress test on Mip-NeRF 360.** We report results on four outdoor scenes with complex geometry, vegetation, and weakly structured regions. **Bold** indicates the best performance, and cell colors highlight the top and second-best.

Method	Mesh	Hybrid	Outdoor: Treehill				Outdoor: Garden				Outdoor: Flowers				Outdoor: Bicycle			
			PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑	PSNR↑	SSIM↑	LPIPS↓	ΔGS↑
3DGS	×	×	20.42	0.684	0.258	0%	28.55	0.883	0.123	0%	**22.48**	**0.641**	0.314	0%	**26.24**	**0.809**	0.187	0%
Hard-Z Proxy	✓	✓	19.21	0.655	0.290	-1%	27.41	0.866	0.119	**7%**	20.77	0.573	0.334	5%	24.33	0.764	0.211	13%
Ours (3DGS)	✓	✓	**20.58**	**0.689**	**0.255**	**14%**	**29.00**	**0.897**	**0.093**	5%	22.34	0.635	**0.295**	**13%**	26.21	0.808	**0.175**	**17%**

hedra, and retain at most the two highest-quality incident tetrahedra for each remaining non-manifold face. The repair is accepted only when the removed fraction is below 0.1% and the final face incidence is at most two. All subsequent MeshSplatting training and rendering stages remain unchanged. Results produced with this fix are identified as the robust-RDel variant.

S2.2. Plug-in Gaussian Backbones

3DGS. The default variant uses a 3DGS checkpoint obtained after the Gaussian-only warm-up. At the hybrid handoff, we retain the Gaussian positions, scales, and rotations, while resetting their color coefficients and opacities. The residual field then continues with the standard 3DGS clone, split, and opacity-based pruning operations. The NDF uses an independent scalar-Gaussian optimizer and densification process.

3DGS-MCMC. For the MCMC variant, the Gaussian-only warm-up is performed using the official 3DGS-MCMC update strategy. Unlike the default 3DGS handoff, the MCMC Gaussian colors and opacities are retained because they determine the current particle distribution and relocation probabilities.

During hybrid optimization, the MCMC operations are applied only to the residual Gaussian field. Low-opacity residual Gaussians are relocated to samples drawn from the alive population, their opacity and scale are adjusted using the original MCMC relocation rule, and new residual Gaussians are added until the prescribed scene capacity is reached. The original opacity-dependent SGLD perturbation is also retained.

The NDF, the textured mesh and continuous fusion remain unchanged.

Because 3DGS-MCMC does not use the opacity-based pruning mechanism of standard 3DGS, we retain its original update strategy, as our goal is to evaluate whether the fusion framework improves the backbone without altering its native optimization behavior. Consequently, this variant can yield slightly negative ΔGS, as reported in the main paper.

Reduced 3DGS. For the Reduced 3DGS variant, the Gaussian-only warm-up is performed using the official Reduced 3DGS update strategy. Because the Gaussolve renderer expects a standard SH tensor layout, the grouped Reduced 3DGS checkpoint is converted to an equivalent full-SH representation at the hybrid handoff. This conversion changes only the storage layout and preserves the Gaussian geometry and appearance.

During hybrid optimization, the redundancy-aware mercy pruning and dead-point removal of Reduced 3DGS are applied only to the residual Gaussian field. Moreover, we retain

its non-DC SH sparsity regularization. At the end of the densification window, multi-view SH-band culling determines the lowest sufficient SH degree for each residual Gaussian and removes the unused higher-order coefficients.

The NDF, the textured mesh and continuous fusion remain unchanged.

Compression codebooks and entropy coding are offline storage mechanisms and are not required for the plug-in training comparison. We therefore transfer the backbone's Gaussian initialization and population-reduction strategy, while storing the final hybrid fields in the common representation used by our renderer.

S3. Surface-Based Dissolve Alternatives

A natural simplification of the Neural Dissolve Field is to directly optimize opacity on the mesh surface. We study two such alternatives: per-vertex alpha and texture-map alpha. Per-vertex alpha is constrained by mesh tessellation and has difficulty isolating thin or localized failure regions. Texture-map alpha is less constrained by vertex count, but the learned pattern is often diffuse and fragmented because texture-space errors do not correspond cleanly to view-space photometric conflicts. By contrast, the volumetric Neural Dissolve Field is localized in 3D and can be view-dependent, allowing the residual Gaussian branch to take over after unreliable mesh evidence is removed.

S3.1. Additional Component Visualizations

Fig. S2 visualizes the mesh, residual, and dissolve components across representative structured and semi-structured scenes. The dissolve field suppresses unreliable mesh evidence, while the residual branch reconstructs the revealed content.

S4. Additional View Consistency

Fig. S3 shows adjacent-view crops under small viewpoint changes. The rendered appearance remains stable around mesh-image conflict regions, indicating that the dissolve-and-refill decomposition does not introduce noticeable view-to-view flicker.

S5. Limitations on Irregular Scenes

Gaussolve is designed for scenes where a lightweight textured mesh can replace broad structured regions, while residual Gaussians handle complex geometry and appearance details. In scenes dominated by foliage, thin branches, and severe geometric ambiguity, the extracted mesh may be too inaccurate to provide a reliable surface prior. The mesh and residual branches may then compete to explain the same image evidence, limiting the reduction in Gaussian primitives.

This defines the intended operating regime of our method: its largest compactness gains arise when broad regular regions can be represented efficiently by the textured mesh. Geometry-aware routing for highly irregular regions is an interesting direction for future work.