arXiv:submit/8327513 [cs.CV] 4 Nov 2026

T^2-Splat: Adaptive Topology Mesh Splatting with Texture Residuals

Zhihao Tang[1] **Youjia Zhang**[1] **Mingbo Zhao**[2] **Wei Yang**[1†]
[1]Huazhong University of Science and Technology
[2]Donghua University

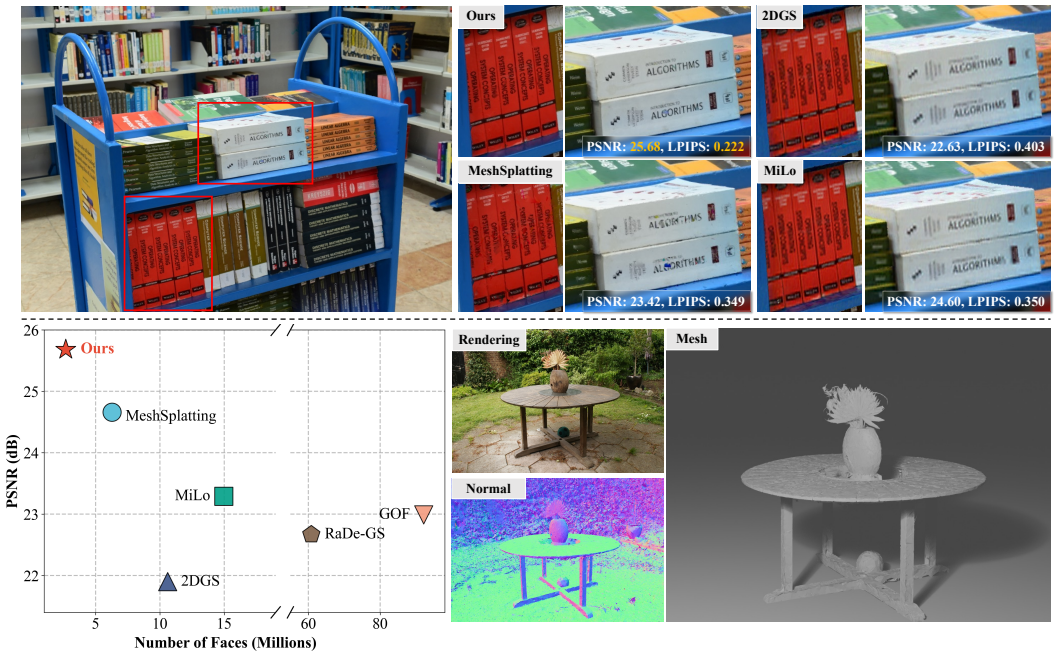


Figure 1: Top: our method better preserves high-frequency details (e.g., text) than MeshSplatting [1] and MiLo [2] on the *shelf* scene (STRinGS-360 [3]). Bottom: on the *garden* scene, T^2-Splat achieves the highest PSNR (25.68 dB) among the compared methods using only 2.7M faces, versus 92M for GOF [4] and 15M for MiLo [2].

Abstract

Triangle-based splatting connects novel view synthesis with conventional graphics pipelines, but often requires excessive tessellation to preserve appearance details. We present T^2-Splat, which decouples topology adaptation from appearance refinement. Safe Restricted Delaunay Triangulation (Safe RDT) initializes mesh connectivity. Dynamic Topology Evolution (DTE) then adapts triangle allocation through edge collapse and vertex split operations. Residual per-face Mip-textures then refine appearance on the fixed mesh using scale-adaptive filtering. Experiments on Mip-NeRF 360, Tanks & Temples, and DTU demonstrate competitive results with compact meshes. T^2-Splat improves PSNR by 0.71 dB over MeshSplatting on Mip-NeRF 360. Additional compression experiments demonstrate flexible control of mesh complexity, supporting face budgets as low as 10% of the original face count. Project page: `https://t2-splat.github.io/`.

†Corresponding author: `weiyangcs@hust.edu.cn`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

3D Gaussian Splatting (3DGS) has enabled high-fidelity and real-time novel view synthesis by representing scenes as sets of anisotropic Gaussian primitives [5]. Despite its impressive rendering quality and optimization efficiency, its unstructured point-based representation remains difficult to integrate into standard graphics pipelines. In particular, 3DGS relies on visibility sorting and alpha compositing, which are not naturally aligned with conventional rasterization mechanisms such as depth buffering, hardware occlusion culling, and mesh-based asset processing [6, 7]. This mismatch complicates deployment in game engines, AR/VR systems, and interactive graphics applications, where triangle meshes remain the dominant representation.

Motivated by this gap, recent work has been resorting to triangle- and mesh-based representations for differentiable novel view synthesis. A line of methods converts Gaussian or neural representations into meshes through post-processing [8, 4, 9, 10], but such two-stage pipelines often decouple rendering optimization from the final mesh structure, leading to detail loss or additional engineering complexity. More recent opaque triangle-based splatting methods directly optimize triangle primitives within differentiable rendering frameworks, improving compatibility with real-time graphics systems while retaining competitive visual quality. For example, MeshSplatting [1] jointly optimizes geometry and appearance with connected opaque meshes and restricted Delaunay triangulation, demonstrating the promise of mesh-native splatting for efficient novel view synthesis. However, existing mesh splatting frameworks still face an important limitation: the allocation of triangle primitives is not sufficiently adaptive to the spatially varying complexity of real scenes.

This limitation arises because triangle primitives carry a dual responsibility. On the one hand, they define the geometric support of the reconstructed scene; on the other hand, they also provide the basis on which appearance features are stored and interpolated. When the mesh contains too many triangles in smooth or textureless regions, the representation becomes unnecessarily redundant and inefficient. Conversely, when the triangle budget is reduced, high-frequency appearance details may be lost, as vertex-based appearance models often require dense tessellation to reproduce fine texture variations. As a result, topology simplification and appearance fidelity become tightly coupled: aggressively collapsing triangles can improve compactness, but may blur textures, introduce aliasing, or degrade view-dependent details. A more desirable mesh splatting representation should therefore adapt its topology according to geometric complexity while providing an appearance model that remains expressive under changing triangle resolutions.

To address this problem, we propose **T^2-Splat**, a mesh splatting framework with adaptive topology and texture optimization. The key idea is to make triangle allocation and appearance modeling mutually supportive rather than restrictive. At the geometry level, we introduce a dynamic topology optimization strategy based on progressive edge-collapse and vertex-split operations [11]. Edge collapse removes redundant triangles in locally simple regions, while vertex split restores degrees of freedom in regions requiring finer geometric or appearance detail. These local and reversible operations allow the mesh to adapt its connectivity during optimization, reducing unnecessary face counts while preserving well-conditioned manifold structure. At the appearance level, we introduce a scale-aware residual mip-texture model. Instead of relying solely on vertex-interpolated appearance features, we attach residual texture signals to triangle primitives and organize them in a mip-texture hierarchy [12]. DTE first adapts mesh connectivity, after which residual per-face textures and vertex SH coefficients are jointly optimized on the fixed mesh to recover appearance details. This separates topology adaptation from subsequent appearance refinement. Moreover, LOD-aware texture sampling provides scale-dependent filtering, reducing aliasing and preserving visual stability across camera distances and viewpoints [13]. Together, adaptive topology evolution and residual mip-texture modeling enable T^2-Splat to achieve a better balance between compact mesh structure and high-fidelity rendering. We evaluate T^2-Splat on the Mip-NeRF 360, Tanks & Temples, and DTU benchmarks. Experiments show that our method improves novel-view rendering quality by +0.71 dB over MeshSplatting on Mip-NeRF 360, while maintaining robust visual fidelity even when the mesh is compressed to 10% of the original face count.

In summary, our main contributions are:

- **T^2-Splat**, a mesh splatting framework featuring a decoupled topology and texture optimization strategy, enabling compact yet high-fidelity novel view synthesis.

- We demonstrate that progressive mesh operations, i.e., edge collapse and vertex split, can be naturally integrated into a differentiable rendering pipeline, for reallocating triangle primitives according to scene complexity while preserving well-conditioned mesh connectivity.
- A scale-aware residual Mip-texture model for triangle primitives that decouples high-frequency appearance from mesh density, improving rendering fidelity under topology simplification.

2 Related Work

Mesh Extraction from Radiance Field. Neural Radiance Fields (NeRF)[14] pioneered photorealistic novel view synthesis, but dense ray sampling incurs high rendering costs. Subsequently, 3D Gaussian Splatting (3DGS)[5] achieved real-time rendering using discrete point-based primitives; however, its unstructured, alpha-blended nature is inherently incompatible with standard graphics pipelines. To bridge this gap, early methods[15, 16, 17, 18, 19] extracted meshes from implicit fields, but this introduced overhead and increased the overall training time. For 3DGS, recent works such as 2DGS[8] and RaDe-GS[9] utilize Truncated Signed Distance Fields (TSDF)[20] for surface extraction, while SuGaR[10] applies Poisson Reconstruction[21] by sampling a surface-aligned Gaussian level set. Additionally, MiLo[2] introduces a hybrid framework utilizing Differentiable Marching Tetrahedra (DMTet)[22] to dynamically recover meshes. However, because these methods still inherently rely on implicit fields or independent post-processing steps , this decoupling between the extraction phase and the optimization phase often leads to a noticeable loss of fine geometric details.

Differentiable Mesh Reconstruction. To avoid precision loss from format conversion, direct end-to-end optimization of meshes has emerged. Early attempts[23, 24] at differentiable mesh reconstruction were primarily limited to simple synthetic objects and often struggled to generalize to real-world complex scenes. Existing methods[25, 26, 27] typically optimize an unstructured collection of triangles, resulting in a lack of connectivity between adjacent triangles. Furthermore, they often fail to generate opaque triangles at the end of training, which limits their usability in downstream applications. While methods like Triangle Splatting+[28] and MeshSplatting[1] achieve connected meshes via Restricted Delaunay Triangulation (RDT)[29] and successfully generate opaque triangles, they suffer from severe Topological Rigidity: once established, the mesh topology is frozen, preventing adaptation to scene complexity. In contrast, T^2-Splat integrates Progressive Meshes into a differentiable pipeline, utilizing explicit edge collapse and vertex split for dynamic geometric complexity adaptation.

Appearance Modeling. Achieving high-fidelity appearance on meshes is challenging. Most mesh-based baselines[26, 1] and early 3DGS methods rely on Spherical Harmonics (SH)[30] attached to vertices. This assumes smooth transitions and severely limits the recovery of high-frequency details. Inspired by recent texture-based Gaussian splatting methods[31, 32, 33, 34, 35], we discard the vertex-only appearance model. Instead, we introduce a Residual Per-face Texture to recover high-frequency spatial details. Furthermore, we integrate Level-of-Detail (LOD)[13] and Mip-texture pyramids[12], enabling scale-adaptive trilinear sampling to effectively eliminate texture aliasing and flickering across viewing distances.

3 Method

The overall architecture of T^2-Splat consists of two main stages. In the geometry stage, we construct an initial manifold mesh using Safe Restricted Delaunay Triangulation (Safe RDT) [29] following MeshSplatting [1] and progressively refine its structure through periodic topology editing. Specifically, we apply edge-collapse and vertex-split operations to adaptively reallocate triangle primitives according to local scene complexity while preserving valid mesh connectivity. In the appearance stage, we introduce zero-initialized residual per-face textures organized in a mip-texture pyramid [12]. This design injects high-frequency appearance details without perturbing the optimized geometry, enabling scale-aware texture filtering and anti-aliased rendering.

3.1 Preliminary

We follow the definition of opaque triangle primitives in MeshSplatting [1], i.e., where each vertex of a triangle stores position v_i, opacity logit a_i, and spherical harmonics coefficients c_i. For a pixel p

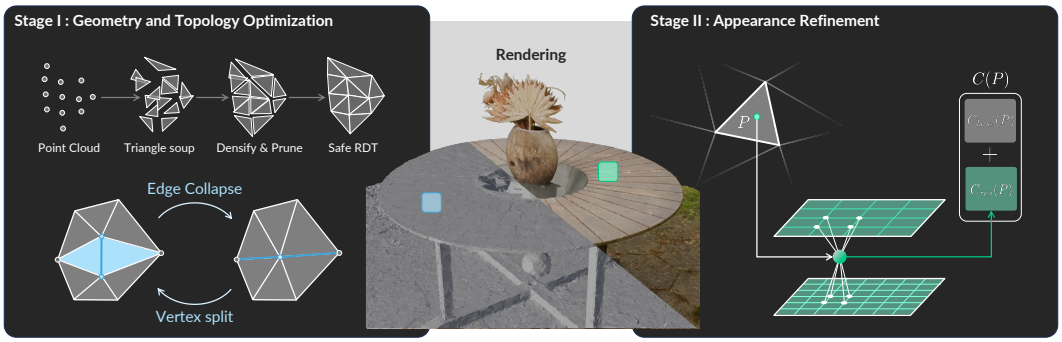


Figure 2: T^2-Splat is an end-to-end differentiable rendering framework that bridges high-fidelity novel view synthesis and standard graphics pipelines. In Stage I, it constructs a manifold mesh from an unstructured triangle soup via Safe Restricted Delaunay Triangulation (Safe RDT), and introduces Dynamic Topology Evolution (DTE) to adaptively optimize topology using QEM-based edge collapses and gradient-driven vertex splits. Stage II employs a scale-aware residual Mip-texture model with trilinear sampling, decoupling appearance from geometry to ensure high-quality, anti-aliased rendering even under extreme mesh compression.

inside a triangle, its color is obtained by barycentric interpolation of vertex attributes:

$$C_{\text{base}}(p,d) = w_i c_i(d) + w_j c_j(d) + w_k c_k(d), \quad w_i + w_j + w_k = 1 \tag{1}$$

where d denotes the viewing direction. While this enables real-time rendering, the spatial resolution of the appearance is strictly tied to the vertex density.

To make triangle rasterization end-to-end differentiable, the discrete triangle indicator is replaced by a soft window function $I(p)$:

$$I(p) = \left(\text{ReLU}\left(\frac{\phi(p)}{\phi(s)}\right)\right)^{\sigma} \tag{2}$$

Here, $\phi(p)$ represents the signed distance field of the 2D triangle in image space, which takes negative values inside, positive outside, and zero on the boundary. The vector s is the incenter of the projected triangle, ensuring that the fraction evaluates to 1 at the center and 0 at the edges.

The exponent σ is a smoothness parameter that controls the transition profile. Concurrently, the rendering opacity α_i is re-parameterized as:

$$\alpha_i = \alpha_{\text{floor}} + (1 - \alpha_{\text{floor}}) \cdot \text{sigm}(a_i) \tag{3}$$

Visible triangles are then alpha-blended in depth order to compute the final baseline pixel color $C_{\text{base}}(p,d)$:

$$C_{\text{base}}(p,d) = \sum_{n=1}^{N} c_{T_n}(p,d)\alpha_{T_n} I(p) \prod_{m=1}^{n-1} (1 - \alpha_{T_m} I(p)) \tag{4}$$

During training, σ is linearly annealed from 1.0 (a smooth gradient facilitating gradient flow) down to 0.0001. As α_{floor} approaches 1 and σ approaches 0, the representation smoothly converges to solid, opaque triangles. Analogously to MobileNeRF [16], this formulation simplifies at the end of training to:$C_{\text{base}}(p,d) = c_{T_n}(p,d)I(p)$,where only a single evaluation per pixel is required. This reduces overdrawing to zero, significantly accelerating the rendering process and enabling native compatibility with standard game engines. This process generates redundant triangles, while geometrically complex regions may lack sufficient degrees of freedom. In the following, we propose to use the progressive mesh operations, i.e., vertex splitting and edge collapsing, to optimize the topology.

3.2 Geometry and Topology Optimization

3.2.1 Safe Restricted Delaunay Triangulation.

To initialize DTE, we augment Restricted Delaunay Triangulation (RDT) with a constrained face-selection procedure, termed *Safe RDT*. Starting from the candidates retained by the RDT intersection

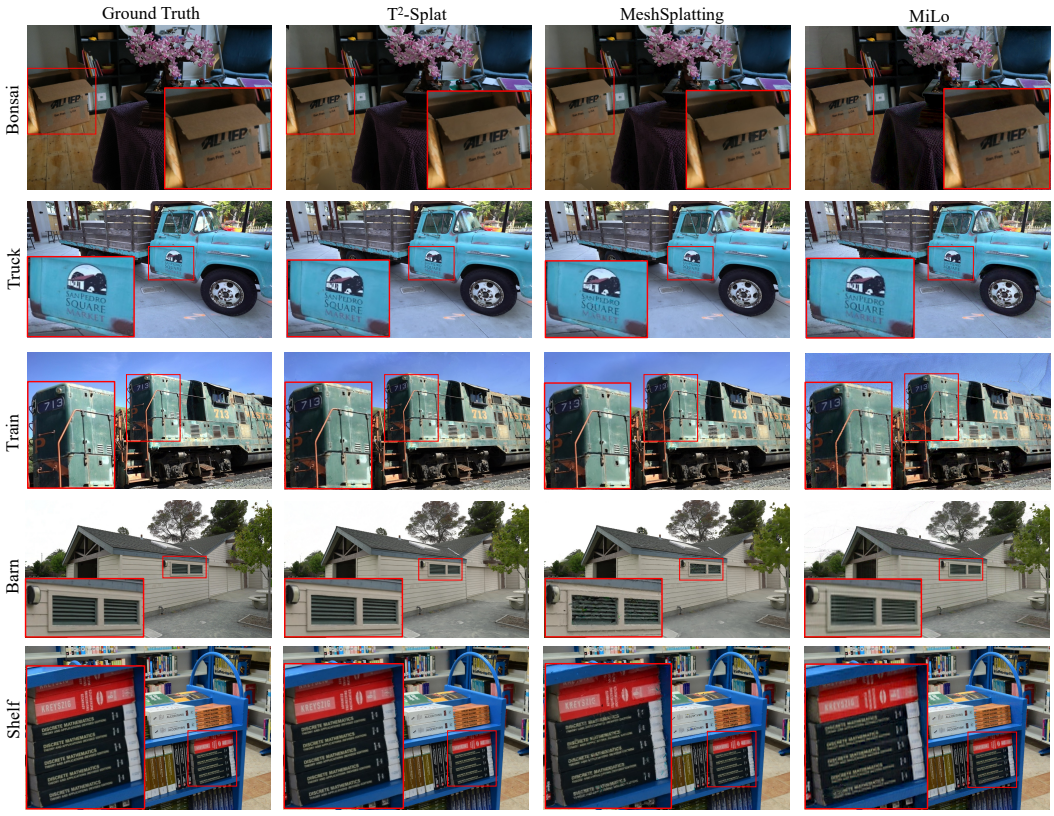


Figure 3: Qualitative results. Compared to baselines that suffer from blurry textures or geometric distortions, our T^2-Splat utilizes per-face textures to effectively eliminate texture distortion and preserves significantly sharper high-frequency details (e.g., fine text and intricate surface patterns) and maintains better structural integrity.

test, we filter and rank faces according to their distance to the input triangle soup, unsigned normal agreement, and triangle quality. We then greedily grow a consistently oriented surface along its edge frontier, ensuring that no duplicate faces are introduced, each edge has at most two incident faces, and each used vertex has a single connected face fan. This procedure updates only face connectivity and winding, retaining vertex positions, appearance and opacity parameters, and their optimizer states. Open boundaries and multiple connected components are permitted. The selected mesh is subsequently refined by DTE through edge collapse and vertex split operations. Further details are provided in the supplementary material.

3.2.2 Edge Collapse: QEM-based Geometric Simplification

To remove redundant triangles in flat regions while preserving boundaries, we use Quadric Error Metrics (QEM). For each face with plane equation $\mathbf{p}_f = [a_f, b_f, c_f, d_f]^T$, its quadric is $K_f = \mathbf{p}_f \mathbf{p}_f^T$. Vertex quadrics are accumulated from adjacent faces (with additional virtual planes for open boundaries):

$$Q_v = \sum_{f \in \mathcal{F}(v)} K_f. \tag{5}$$

For a candidate edge $e = (u, v)$, the merged quadric is $Q_{uv} = Q_u + Q_v$. The optimal new vertex position minimizes the homogeneous quadric error:

$$\bar{v}^* = \arg\min_{\bar{v}} \bar{v}_h^T Q_{uv} \bar{v}_h, \quad \bar{v}_h = [x, y, z, 1]^T. \tag{6}$$

If the system is singular, we fall back to the best discrete candidate:

$$\bar{v}^* = \arg \min_{c \in \{u, v, (u+v)/2\}} c_h^T Q_{uv} c_h. \tag{7}$$

The collapse cost combines the normalized QEM error with base-color and activated opacity differences to discourage collapses across appearance and opacity discontinuities:

$$\mathrm{Cost}(e) = w_q \frac{(\bar{\mathbf{v}}_h^*)^\top Q_{uv} \bar{\mathbf{v}}_h^*}{\mathrm{median}(E_q) + \epsilon} + w_c \left\| \mathbf{c}_u^{\mathrm{dc}} - \mathbf{c}_v^{\mathrm{dc}} \right\|_2 + w_a \left| \alpha_u - \alpha_v \right|. \tag{8}$$

Here, α_i denotes the activated vertex opacity defined in Eq. (3) and w_a controls the opacity term. After sorting candidate edges by cost, we select collapses subject to a maximum edge-length threshold and the full simplicial link condition:

$$\mathrm{Lk}_{\mathcal{M}}(u) \cap \mathrm{Lk}_{\mathcal{M}}(v) = \mathrm{Lk}_{\mathcal{M}}(\{u, v\}). \tag{9}$$

Here, $\mathrm{Lk}_{\mathcal{M}}(\cdot)$ denotes the simplicial link in the current mesh $\mathcal{M}$, including its vertices and edges. We additionally reject collapses that join two boundary vertices through an interior edge, produce an invalid merged vertex fan, or introduce local face degeneracy or orientation reversal. Accepted collapses within each update have disjoint affected one-ring neighborhoods.

Appearance attributes for the new vertex are merged via opacity-weighted averaging:

$$c_{new} = \frac{\alpha_u c_u + \alpha_v c_v}{\alpha_u + \alpha_v + \epsilon}. \tag{10}$$

3.2.3 Vertex Splitting: Gradient-driven Refinement

To increase local mesh density in complex regions, we calculate a perspective-compensated gradient score using the position gradient magnitude and camera-space depth d_i:

$$g_i^c = \|\nabla_{v_i} \mathcal{L}\| \cdot d_i. \tag{11}$$

This score is accumulated and normalized across N_i training views:

$$\bar{g}_i^c = \frac{1}{N_i} \sum_{k=1}^{N_i} g_{i,k}^c. \tag{12}$$

Vertices with high $\bar{g}_i^c$ are refined by inserting a new vertex at the midpoint of their longest adjacent edge:

$$v_{new} = \frac{1}{2}(v_i + v_j). \tag{13}$$

The incident triangle is replaced by two new triangles while preserving orientation:

$$(i, j, k) \to (i, v_{new}, k), (v_{new}, j, k). \tag{14}$$

Finally, the attributes of the inserted vertex are linearly interpolated:

$$c_{new} = \frac{c_i + c_j}{2}, \quad a_{new} = \frac{a_i + a_j}{2}. \tag{15}$$

3.2.4 Optimizer State Stitching

To preserve optimization history across edge collapse and vertex split operations, we transfer Adam's first and second moments, denoted by m and v, to the updated parameters.

During edge collapse, the merged vertex inherits the moments of the two endpoints through incident-area-weighted averaging:

$$m_{\mathrm{new}} = \frac{A_u m_u + A_v m_v}{A_u + A_v}, \qquad v_{\mathrm{new}} = \frac{A_u v_u + A_v v_v}{A_u + A_v}, \tag{16}$$

where A_u and A_v are the incident-area weights of vertices u and v.

During vertex splitting, the retained parent k and the newly inserted midpoint vertex k' inherit the parent's moments, with opposite perturbations applied to the first moment:

$$\begin{aligned} m_k^+ &= m_k + \Delta m, \qquad m_{k'} = m_k - \Delta m, \\ v_k^+ &= v_{k'} = v_k. \end{aligned} \tag{17}$$

Here, $+$ denotes the updated state of the retained vertex, $\Delta m = 10^{-3}\sqrt{v_k} \odot z$, and $z \sim \mathcal{N}(0, I)$. The square root and multiplication are element-wise.

Table 1: **Mesh-based NVS results on Mip-NeRF360 and Tanks & Temples.** We report PSNR, SSIM, LPIPS, and vertex count. *Mesh*, *Color*, and *Connect* indicate whether the method directly outputs a mesh, whether the mesh carries color without post-processing, and whether the output is connected, respectively.

Method	Characteristics			Mip-NeRF360 dataset				Tanks & Temples			
	Mesh	Color	Connect	PSNR ↑	LPIPS ↓	SSIM ↑	#Verts ↓	PSNR ↑	LPIPS ↓	SSIM ↑	#Verts ↓
3DGS	–	–	–	27.21	0.214	0.815	–	23.14	0.183	0.841	–
Triangle Splatting	–	–	–	27.16	0.191	0.814	–	23.14	0.143	0.857	–
2DGS	✗	✗	✓	15.36	0.474	0.498	2M	14.23	0.485	0.569	16M
GOF	✗	✗	✓	20.78	0.465	0.573	33M	21.69	0.326	0.690	12M
RaDe-GS	✗	✗	✓	23.56	0.361	0.668	31M	20.51	0.344	0.659	10M
MiLo	✓	✗	✓	24.09	0.323	0.688	7M	21.46	0.348	0.706	4M
Gaussian Wrapping	✓	✗	✓	25.10	0.289	0.752	12M	20.98	0.367	0.700	6M
Triangle Splatting	✓	✓	✗	21.05	0.462	0.558	3M	17.27	0.402	0.600	6M
MeshSplatting	✓	✓	✓	24.78	0.310	0.728	3M	20.52	0.287	0.745	2M
T^2-Splat	✓	✓	✓	25.49	0.234	0.779	2M	21.12	0.213	0.795	1M

Table 2: **Chamfer distance on the DTU dataset.** We report Chamfer distance across 15 scenes. T^2-Splat achieves a joint second-best mean Chamfer distance of 0.68 and the lowest error on three scenes (63, 83, and 105).

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
3DGS	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	2.22	1.75	1.79	2.55	1.53	1.52	1.50	1.96
2DGS	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80
GOF	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	1.29	0.68	0.77	0.90	0.42	0.66	0.49	0.74
RaDe-GS	0.46	0.73	0.33	0.38	0.79	0.75	0.76	1.19	1.22	0.62	0.70	0.78	0.36	0.68	0.47	0.68
MiLo	0.43	0.74	0.34	0.37	0.80	0.74	0.70	1.21	1.22	0.66	0.62	0.80	0.37	0.76	0.48	0.68
Triangle Splatting	0.98	1.07	1.07	0.51	1.67	1.44	1.17	1.32	1.75	0.98	0.96	1.11	0.56	0.93	0.72	1.08
MeshSplatting	0.77	0.72	0.87	0.86	0.81	1.35	0.89	1.08	1.15	0.58	0.95	1.00	0.75	0.86	0.78	0.89
Gaussian Wrapping	0.40	0.51	0.27	0.32	0.81	0.46	0.45	1.04	0.65	0.52	0.33	0.59	0.32	0.33	0.34	0.49
T^2-Splat	0.56	0.63	0.55	0.47	0.77	1.09	0.69	0.98	1.18	0.47	0.63	0.75	0.47	0.53	0.48	0.68

3.3 Textural Residual Model

With well-conditioned topology, we freeze the mesh topology and vertex positions to focus on appearance refinement. While vertex-based Spherical Harmonics (SH) capture smooth view-dependent effects, they struggle with high-frequency details. We therefore introduce a learnable residual texture $T_f \in \mathbb{R}^{R\times R\times 3}$ for each face f, initialized to zero to ensure a lossless transition from Stage I.

3.3.1 Residual Parameterization and Mirror Mapping

For a pixel p within face f, we define local texture coordinates (u, v) via its barycentric coordinates (w_i, w_j, w_k) such that $u = w_j, v = w_k$. The final color $C(p, d)$ integrates vertex-based shading and per-face residual sampling:

$$C(p,d) = \underbrace{C_{base}(p,d)}_{\text{View-dependent Base}} + \underbrace{C_{res}(u,v)}_{\text{Spatial Residual Texture}}, \tag{18}$$

where d denotes the viewing direction and $C_{res}(u, v) = \text{Bilinear}(T_f, u, v)$. To efficiently utilize the square texture domain and stabilize boundary interpolation, we employ *mirror mapping*: if $u+v > 1$, the coordinates are reflected back into the valid triangular domain via $T_f(u, v) = T_f(1 - v, 1 - u)$. This symmetry prevents sampling artifacts and ensures gradient consistency during optimization.

3.3.2 Scale-adaptive Trilinear Sampling

To mitigate aliasing caused by small texture footprints, we construct a differentiable Mip-texture pyramid where coarser levels are generated by average pooling: $T_f^{l+1} = \text{AvgPool}(T_f^l)$. The level of

Table 3: **OpenGL rendering efficiency.** All methods are evaluated on the Garden scene using the same OpenGL pipeline and rendering configuration at a resolution of 1920×1080 on a MacBook Air M3. Storage reports mesh file size on disk. OOM denotes an out-of-memory failure. Our method achieves the highest frame rate and smallest storage footprint.

Method	FPS ↑	Storage ↓	#Faces ↓
GOF	OOM	1.8 GB	92.1M
RaDe-GS	OOM	1.1 GB	60.8M
MiLo	82	314 MB	14.9M
MeshSplatting	126	152 MB	7.3M
Gaussian Wrapping	40	638 MB	31.2M
Ours	**236**	**62 MB**	**2.7M**

Table 4: **Quantitative evaluation under varying mesh counts on the Garden scene.** Percentages in parentheses represent the ratio of retained mesh faces. Compared to MeshSplatting, T^2-Splat achieves robust visual fidelity even when the mesh size is aggressively reduced (e.g., to 10%).

Method	PSNR ↑	LPIPS ↓	SSIM ↑	#Faces ↓
RaDe-GS	22.67	0.21	0.743	60.8M
GOF	22.99	0.26	0.706	92.1M
2DGS	21.90	0.32	0.608	10.6M
MiLo	23.29	0.22	0.738	14.9M
MeshSplatting	24.70	0.22	0.762	7.3M
Gaussian Wrapping	23.45	0.27	0.717	31.2M
Ours (100%)	**25.68**	**0.15**	**0.818**	**2.7M**
Ours (50%)	**25.38**	**0.17**	**0.801**	**1.4M**
Ours (10%)	**23.93**	**0.30**	**0.697**	**0.3M**

detail λ is estimated from screen-space derivatives:

$$\lambda = \log_2 \left(\max \left(\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} \right) + \epsilon \right). \tag{19}$$

Defining $D = \lfloor \lambda \rfloor$ and $\delta = \lambda - D$, the final texture value $C^{\lambda}_{res}(u, v)$ is computed via trilinear interpolation between adjacent Mip levels:

$$C^{\lambda}_{res}(u, v) = (1 - \delta)\,\mathrm{Bilinear}(T^{D}_{f}, u, v) + \delta\,\mathrm{Bilinear}(T^{D+1}_{f}, u, v). \tag{20}$$

The total color is thus refined as $C(p, d) = C_{base}(p, d) + C^{\lambda}_{res}(u, v)$, effectively acting as a dynamic low-pass filter to eliminate flickering while preserving high-frequency details across varying scales.

3.4 Optimization Objectives

In the geometric stage, we optimize vertex positions v_i, opacities o_i, and spherical harmonic (SH) coefficients c_i with a joint objective:

$$\mathcal{L} = \mathcal{L}_{\text{3DGS}} + \beta_o \mathcal{L}_o + \beta_z \mathcal{L}_z + \beta_n \mathcal{L}_n + \beta_d \mathcal{L}_d. \tag{21}$$

Here, $\mathcal{L}_{\text{3DGS}}$ combines the photometric $\mathcal{L}_1$ and $\mathcal{L}_{\text{D-SSIM}}$ losses from 3DGS [5], while $\mathcal{L}_o$ regularizes vertex opacity [36]. The vertex depth alignment loss penalizes the deviation between each vertex depth z_i and the sampled rendered depth z_i^*:$\mathcal{L}_z = \frac{1}{N}\sum_{i=1}^{N} |z_i - z_i^*|$. The normal loss $\mathcal{L}_n$ enforces consistency between rendered and surface normals, using supervision from external normal estimators [37, 38], 2DGS self-regularization [8], or both. The depth loss $\mathcal{L}_d$ further aligns predicted depths with monocular priors from Depth Anything v2 [39] after scale-and-shift normalization [40].

In the appearance stage, we freeze mesh connectivity, vertex positions, and opacities, and jointly optimize the residual per-face textures and vertex SH coefficients. The texture objective is defined as

$$\mathcal{L}_{\text{texture}} = \lambda_{l1}\mathcal{L}_1 + \lambda_{ssim}\mathcal{L}_{\text{SSIM}} + \lambda_{tv}\mathcal{L}_{\text{TV}} + \lambda_{l2}\mathcal{L}_{\text{L2}}. \tag{22}$$

The TV loss smooths adjacent texels, while the $\mathcal{L}_2$ term constrains the residual magnitude, ensuring that per-face textures mainly compensate for high-frequency details. Together with real-time Mip-texture pyramids and trilinear sampling, this stage improves multi-scale rendering quality and reduces distant-view aliasing.

4 Experiments

In this section, we compare our proposed method with existing approaches on the Mip-NeRF360[41], Tanks & Temples[42], and DTU[43] datasets. Following MiLo[2] and MeshSplatting[1], our evaluation is primarily based on the task of mesh-based novel view synthesis, which measures the

Table 5: **Ablations on Mip-NeRF 360.** We progressively enable Safe RDT, Dynamic Topology Evolution (DTE), and residual per-face Mip-textures (MipTex) from baseline (a). DTE improves rendering quality with fewer faces, while Mip-textures further refine appearance on fixed geometry. Results are averaged equally over nine scenes.

	Safe RDT	DTE	MipTex	PSNR ↑	LPIPS ↓	SSIM ↑	#Faces ↓	Time (min) ↓
(a)	–	–	–	24.66	0.30	0.734	5.3M	54
(b)	✓	–	–	24.81	0.31	0.740	3.0M	43
(c)	✓	✓	–	25.12	0.29	0.756	**2.5M**	57
(d)	✓	–	4×4	25.20	0.27	0.762	3.0M	65
(e)	✓	–	8×8	25.31	0.24	0.769	3.0M	81
(f)	✓	✓	4×4	25.43	0.26	0.775	**2.5M**	72
(g)	✓	✓	8×8	**25.49**	**0.23**	**0.779**	**2.5M**	83

performance of reconstructed meshes in complex real-world scenes. Additionally, we calculate the Chamfer Distance (CD) on the DTU dataset to evaluate the accuracy of our mesh reconstruction.

Baselines. We compare our method against 2DGS[8], MiLo[2], GOF[4], RaDe-GS[9], Gaussian Wrapping[44], Triangle Splatting[26], and MeshSplatting[1]. Note that 2DGS, GOF and RaDe-GS require post-processing steps for mesh extraction, whereas MiLo integrates mesh extraction into the optimization process itself. However, all these methods require training a neural color field to texture the mesh (refer to MiLo for detailed procedures). In contrast, Triangle Splatting and MeshSplatting utilize end-to-end mesh training and do not require post-hoc extraction or coloring. Qualitative comparisons demonstrate the results of our method against state-of-the-art baselines, MiLo and MeshSplatting, on the Mip-NeRF360 and Tanks & Temples datasets.

Implementation Details. All training experiments are conducted on a single NVIDIA RTX 6000 Ada GPU (48GB). Each training run consists of 35,000 iterations, with 30,000 iterations dedicated to geometric optimization and 5,000 iterations for texture refinement. More details are provided in the supplementary material.

4.1 Mesh-based NVS and Surface Reconstruction

We report the quantitative results for mesh-based NVS on the Mip-NeRF360 and Tanks & Temples datasets in Table 1. On Mip-NeRF 360, our method significantly outperforms the state-of-the-art MeshSplatting in terms of PSNR, LPIPS, and SSIM, while utilizing fewer vertices and faces. This demonstrates that our method achieves an optimal balance between rendering quality and geometric reconstruction, particularly in capturing high-frequency details. On Tanks & Temples, our method achieves the best LPIPS and SSIM among the evaluated mesh-based renderings, while GOF and MiLo obtain higher PSNR. Our method, while outperforming MeshSplatting in PSNR, significantly surpasses all mesh-based baselines in LPIPS and SSIM, proving its superior structural and visual quality, especially in terms of realistic detail recovery. Qualitative results are shown in Figure 3.

We report the Chamfer Distance for each method across 15 scenes of the DTU dataset in Table 2. Our method achieves the lowest Chamfer Distance in 3 scenes, which indicates that although our framework is primarily targeted at end-to-end mesh reconstruction and rendering for complex real-world scenes, it still achieves surface reconstruction quality comparable to concurrent methods. Furthermore, our method outperforms MeshSplatting in most scenes, demonstrating that the DTE module—through edge collapse and vertex splitting—effectively improves mesh quality.

4.2 Rendering Efficiency and Storage

Table 3 compares OpenGL rendering efficiency on the Garden scene using a MacBook Air M3 under the same rendering configuration. Our method achieves 236 FPS, approximately $1.87\times$ the frame rate of MeshSplatting (126 FPS), while reducing the mesh file size from 152 MB to 62 MB, a reduction of approximately 59%. GOF and RaDe-GS exceed the available memory under this setup. These results show that our compact mesh representation supports efficient rendering with reduced mesh

storage requirements in a standard OpenGL pipeline. For our method, storage and FPS refer to the baked vertex-colored PLY representation described in the supplementary material.

4.3 Extreme Mesh Simplification and Robustness Analysis

As shown in Table 4, we evaluate rendering quality under varying mesh scales on the Garden scene. While existing methods demand massive face counts (e.g., GOF's 92.1M and MiLo's 14.9M) to maintain fidelity, our full model achieves state-of-the-art results across all metrics (PSNR 25.68, LPIPS 0.15) with only 2.7M faces.

To evaluate rendering quality under different mesh budgets, we control the final face count by adjusting the relative rates of vertex split and edge collapse in DTE during Stage I. Each resulting mesh is then held fixed during Stage II, where residual per-face textures and vertex SH coefficients are optimized. On Garden, our 50% configuration achieves higher PSNR, lower LPIPS, and higher SSIM than MeshSplatting, using only 1.4M faces compared with 7.3M. Further reducing the budget to 10% yields 0.3M faces, with 23.93 dB PSNR, 0.30 LPIPS, and 0.697 SSIM. This demonstrates the trade-off between rendering quality and geometric compactness under increasingly limited mesh budgets.

4.4 Ablations

We evaluate the contributions of Safe RDT, Dynamic Topology Evolution (DTE), and residual per-face Mip-textures on Mip-NeRF 360. Table 5 reports results averaged over nine scenes.

Safe RDT. Comparing configurations (a) and (b), Safe RDT reduces the average face count from 5.3M to 3.0M, a reduction of approximately 43%, while increasing PSNR from 24.66 to 24.81 dB and SSIM from 0.734 to 0.740. LPIPS increases slightly from 0.30 to 0.31. These results show that constrained RDT face selection provides a more compact initialization while maintaining comparable rendering quality.

Dynamic Topology Evolution. Adding DTE to Safe RDT, as shown by (b) versus (c), improves PSNR by 0.31 dB while further reducing the face count from 3.0M to 2.5M. DTE also improves rendering quality when combined with per-face textures: comparing (d) with (f) and (e) with (g), it increases PSNR by 0.23 and 0.18 dB for 4×4 and 8×8 textures, respectively. These consistent improvements support the effectiveness of adapting mesh connectivity through edge collapse and vertex split operations.

Residual Per-face Mip-textures. Appearance refinement improves rendering fidelity on fixed geometry. Starting from the DTE-refined mesh in (c), adding 4×4 textures in (f) increases PSNR from 25.12 to 25.43 dB. Increasing the texture resolution to 8×8 in (g) further improves PSNR to 25.49 dB, reduces LPIPS from 0.26 to 0.23, and increases SSIM from 0.775 to 0.779, without changing the face count. The higher resolution increases training time from 72 to 83 minutes, illustrating the trade-off between appearance quality and optimization cost. The complete configuration (g) improves PSNR by 0.83 dB over baseline (a) and reduces the face count by approximately 53%, while increasing total training time from 54 to 83 minutes.

5 Conclusions

We presented T^2-Splat, a two-stage framework that combines adaptive mesh topology with residual textures for high-fidelity rendering. Safe RDT initializes mesh connectivity under local manifold constraints, while Dynamic Topology Evolution (DTE) adapts geometric complexity through edge collapses and vertex splits. After geometry optimization, vertex SH coefficients and residual per-face Mip-textures are jointly optimized on the fixed mesh to recover fine appearance details, with scale-adaptive filtering reducing aliasing. Experiments across multiple datasets demonstrate improved rendering quality with compact geometry and competitive surface reconstruction accuracy. Controlling the relative rates of topology operations further provides a flexible trade-off between rendering quality and mesh size. The resulting meshes can be exported in PLY and GLB formats for use in standard graphics pipelines.

Acknowledgments and Disclosure of Funding

We thank the reviewers for their valuable comments and constructive suggestions, which helped improve this paper. We also thank the authors and maintainers of the publicly available datasets and open-source implementations used in this work for supporting the development and evaluation of our method. This work is supported by the National Natural Science Foundation of China (NSFC Grant No. 62272184).

References

[1] Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. G Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. Meshsplatting: Differentiable rendering with opaque meshes. *arXiv*, 2025.

[2] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. Milo: Mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics (TOG)*, 44(6):1–15, 2025.

[3] Abhinav Raundhal, Gaurav Behera, P. J. Narayanan, Ravi Kiran Sarvadevabhatla, and Makarand Tapaswi. Strings: Selective text refinement in gaussian splatting. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, March 2026.

[4] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024.

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[6] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019.

[7] James D Foley. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996.

[8] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657428.

[9] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiao-Xiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *ACM Transactions on Graphics*, 45(2):1–14, 2026.

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5354–5363, 2024.

[11] H HOPPE. Progressive meshes. In *Proceedings of ACM SIGGRAPH 96, Computer Graphics*, pages 99–108, 1996.

[12] Jon P Ewins, Marcus D Waller, Martin White, and Paul F Lister. Mip-map level selection for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):317–329, 1998.

[13] David Luebke, Martin Reddy, Jonathan D Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. *Level of detail for 3D graphics*. Elsevier, 2002.

[14] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65 (1):99–106, 2021.

[15] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–9, 2023.

[16] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023.

[17] Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

[18] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in neural information processing systems*, 34:4805–4815, 2021.

[19] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[20] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[21] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[22] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[23] Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C Lin, and Yi Zhou. Dmesh: A differentiable mesh representation. *arXiv preprint arXiv:2404.13445*, 2024.

[24] Sanghyun Son, Matheus Gadelha, Yang Zhou, Matthew Fisher, Zexiang Xu, Yi-Ling Qiao, Ming C Lin, and Yi Zhou. Dmesh++: An efficient differentiable mesh for complex shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26590–26599, 2025.

[25] Changjian Jiang, Kerui Ren, Linning Xu, Jiong Chen, Jiangmiao Pang, Yu Zhang, Bo Dai, and Mulin Yu. Halogs: Loose coupling of compact geometry and gaussian splats for 3d scenes. *arXiv preprint arXiv:2505.20267*, 2025.

[26] Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. *arXiv*, 2025.

[27] Nathaniel Burgdorfer and Philippos Mordohai. Radiant triangle soup with soft connectivity forces for 3d reconstruction and novel view synthesis. *arXiv preprint arXiv:2505.23642*, 2025.

[28] Jan Held, Renaud Vandeghen, Sanghyun Son, Daniel Rebain, Matheus Gadelha, Yi Zhou, Ming C Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. Triangle splatting+: Differentiable rendering with opaque triangles. *arXiv preprint arXiv:2509.25122*, 2025.

[29] Siu-Wing Cheng, Tamal Krishna Dey, Jonathan Shewchuk, and Sartaj Sahni. *Delaunay mesh generation*. CRC Press Boca Raton, 2013.

[30] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference*, volume 56, page 4, 2003.

[31] Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N Kutulakos, and David B Lindell. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. *arXiv preprint arXiv:2409.12954*, 2024.

[32] David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. Billboard splatting (bbsplat): Learnable textured primitives for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25029–25039, 2025.

[33] Yiming Wang, Shaofei Wang, Marko Mihajlovic, and Siyu Tang. Neural texture splatting: Expressive 3d gaussian splatting for view synthesis, geometry, and dynamic reconstruction. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

[34] Tian-Xing Xu, Wenbo Hu, Yu-Kun Lai, Ying Shan, and Song-Hai Zhang. Texture-gs: Disentangling the geometry and texture for 3d gaussian splatting editing. In *European Conference on Computer Vision*, pages 37–53. Springer, 2024.

[35] Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, and Changil Kim. Textured gaussians for enhanced 3d scene appearance modeling. In *CVPR*, 2025.

[36] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.

[37] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9043–9053, 2023.

[38] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10579–10596, 2024.

[39] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

[40] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions On Graphics (TOG)*, 43(4):1–15, 2024.

[41] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[42] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[43] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[44] Diego Gomez, Antoine Guédon, Nissim Maruani, Bingchen Gong, and Maks Ovsjanikov. From blobs to spokes: High-fidelity surface reconstruction via oriented gaussians. In *European Conference on Computer Vision*, pages 54–72. Springer, 2026.

A Supplementary Material

This supplement expands the implementation and analysis of T^2-Splat. Section A.1 describes the training schedule and parameter groups; Sections A.2 and A.3 detail mesh initialization and topology updates; Section A.4 explains residual texture coordinates and filtering; Section A.5 distinguishes the trained representation from exported assets; and Sections A.6 and A.7 discuss additional results and limitations.

A.1 Training Schedule and Configuration

All training experiments use a single NVIDIA RTX 6000 Ada GPU with 48 GB of memory. Each run contains 35,000 iterations: 30,000 iterations for geometry and base appearance optimization, followed by 5,000 iterations of appearance refinement. Table 6 lists the reported hyperparameters, and Table 7 separates the roles of the different training intervals.

Table 6: Main hyperparameters used in our experiments.

	Outdoor	Indoor
feature learning rate	0.0016	0.004
opacity learning rate	0.03	0.05
vertex pos. learning rate	0.0015	0.0015
texture learning rate	0.0005	0.0005
densification start	500	500
densification end	10000	10000
densification interval	500	500
start pruning	4000	3000
pruning threshold	0.235	0.235
mesh creation	11000	11000
DTE start	12000	12000
DTE end	20000	20000
β_z	0.00025	0.00025
β_n	0.0001	0.0001
β_d	0.01	0.0
β_o	2e-06	0.0
λ_{l1}	0.8	0.8
λ_{ssim}	0.2	0.2
λ_{tv}	0.01	0.01
λ_{l2}	0.01	0.01

Table 7: Training stages and their roles. Iteration indices describe the reported schedule; the beginning and end of each interval mark stage transitions.

Iteration / event	Representation update	Optimization role
500–10,000	Densification every 500 iterations	Establish geometric support before mesh construction; pruning starts at 4,000 outdoors and 3,000 indoors.
11,000	Safe RDT	Replace soup connectivity with a constrained face set; retain vertex parameters and optimizer states.
12,000–20,000	DTE every 1,000 iterations	Adapt connectivity using edge collapse and gradient-driven refinement while optimizing geometry and base appearance.
20,000–30,000	DTE interval completed	Continue Stage I optimization of vertex positions, opacities, and SH coefficients.
30,000–35,000	Fixed topology, positions, opacity	Jointly optimize vertex SH coefficients and residual per-face textures.

Topology update budget. Within the DTE interval, edge collapse targets 5%–10% of the current vertices, whereas refinement targets the top 5% ranked by perspective-compensated position gradients. Local validity constraints determine which candidate edits are accepted. The resulting face-count change depends on the local neighborhood and whether the edited edge is on a boundary. The separate 50% and 10% configurations in Table 4 refer to retained face budgets relative to the full Garden configuration, obtained by adjusting the relative rates of the two operations.

Appearance-stage parameter groups. Residual textures use 8×8 RGB grids by default and are initialized to zero. Their base learning rate is 5×10^{-4} with exponential decay; the reported learning rate for higher-order SH coefficients during refinement is 2×10^{-4}. Geometry and opacity parameters remain fixed in this stage. The 4×4 variant in Table 5 changes texture resolution, with geometry held fixed within each appearance-refinement run. The appearance objective uses weights 0.8, 0.2, 0.01, and 0.01 for the photometric $\mathcal{L}_1$, SSIM-based, total-variation, and residual-magnitude terms, respectively. The latter two terms discourage oscillatory texels and unnecessarily large residual corrections.

A.2 Safe RDT: Candidate Selection and Local Connectivity

To construct a surface mesh from the optimized triangle soup, we extend the Restricted Delaunay Triangulation (RDT) procedure used in MeshSplatting [1]. We first perform Delaunay tetrahedralization on the optimized vertex set using SciPy and construct dual segments by connecting the circumcenters of adjacent tetrahedra. A Bounding Volume Hierarchy (BVH) accelerates intersection queries against the input triangle soup, with Möller–Trumbore tests performed at the leaf nodes. BVH traversal and intersection testing are accelerated using Taichi. Internal tetrahedral faces whose dual segments intersect the input surface form the candidate face set.

We further introduce *Safe RDT* to filter these candidates and select a consistently oriented surface. For each candidate face f, let $\mathbf{c}_f$, $\mathbf{n}_f$, and A_f denote its centroid, unit normal, and area. Let $\mathbf{p}_f$ be the closest point on the nondegenerate triangles of the input soup, and let g_f denote the corresponding input triangle. We evaluate

$$\begin{aligned} d_f &= \frac{\|\mathbf{c}_f - \mathbf{p}_f\|_2}{\max(\ell_{\max}(g_f), \epsilon)}, \\ a_f &= \left|\mathbf{n}_f^\top \mathbf{n}_{g_f}\right|, \\ q_f &= \frac{4\sqrt{3}A_f}{\sum_{e \in E(f)} \ell_e^2}, \end{aligned} \tag{23}$$

where $\ell_{\max}(g_f)$ is the longest edge of g_f, and ℓ_e is the length of an edge in $E(f)$. These quantities measure normalized surface distance, unsigned normal agreement, and triangle quality, respectively. We retain nondegenerate candidates satisfying $d_f \le 0.5$, $a_f \ge 0.15$, and $q_f \ge 0.02$, and rank them using

$$s_f = a_f \exp(-4d_f)\frac{1 + q_f}{2}. \tag{24}$$

Starting from an empty face set, we greedily grow the surface through a score-prioritized edge frontier. Candidate insertion must introduce no duplicate face, keep each edge incident to at most two faces, preserve a single path or cycle in the link of every used vertex, and admit consistent face orientations. Candidates that temporarily violate the vertex-fan condition are deferred and reconsidered as neighboring faces are accepted. A new component is seeded when the frontier is exhausted. Orientation consistency is maintained using a disjoint-set structure with parity information.

Safe RDT updates only face connectivity and winding, retaining vertex positions, opacity and SH parameters, and their optimizer states. It permits open boundaries and multiple connected components. Geometric filtering favors local fidelity, while the topology constraints maintain valid surface neighborhoods. The selected mesh initializes DTE, which further adapts its connectivity through edge collapse and vertex split operations.

Interpretation of the geometric criteria. The distance score d_f is dimensionless: normalizing by the local longest edge compares displacement relative to the support scale of the input soup. The absolute dot product a_f measures normal agreement independently of input winding, while

the connectivity-selection procedure determines consistent face orientations. For a nondegenerate triangle, q_f lies in $(0, 1]$, equals one for an equilateral triangle, and tends to zero as the triangle degenerates. It is invariant to uniform scaling because both area and squared edge length scale quadratically. Thus the three tests penalize different failure modes: distance from the input surface, incompatible local normals, and poorly shaped triangles.

Joint edge and vertex constraints. Edge-incidence constraints limit each edge to at most two incident faces. Vertex-link constraints additionally require a single path or cycle around each used vertex, corresponding to boundary or interior surface neighborhoods. Together, these conditions exclude disconnected face fans at a shared vertex. Deferral allows a temporarily disconnected candidate fan to become connected as neighboring faces are accepted during greedy surface growth.

A.3 Dynamic Topology Evolution and Optimizer Continuity

QEM minimization in Cartesian coordinates. For an edge (u, v), write the merged quadric in block form as

$$Q_{uv} = \begin{bmatrix} H & \mathbf{b} \\ \mathbf{b}^\top & c \end{bmatrix}, \qquad E(\mathbf{x}) = \mathbf{x}^\top H\mathbf{x} + 2\mathbf{b}^\top\mathbf{x} + c. \tag{25}$$

Differentiating gives $\nabla E(\mathbf{x}) = 2H\mathbf{x} + 2\mathbf{b}$. When H is invertible, the unconstrained minimizer satisfies $H\mathbf{x}^* = -\mathbf{b}$. For a singular system, the endpoint and midpoint fallback in the main paper selects the lowest-error member of $\{\mathbf{u}, \mathbf{v}, (\mathbf{u} + \mathbf{v})/2\}$. The minimizer supplies a candidate location, which is evaluated using the appearance-aware cost and validity checks.

Boundary protection as an additional quadric. The standard boundary construction uses a plane through the edge and perpendicular to its incident face. For an open boundary edge, let $\mathbf{t}$ be its unit tangent and $\mathbf{n}_f$ the normal of its incident face. A plane containing the edge and perpendicular to that face has unit normal

$$\mathbf{n}_b = \frac{\mathbf{t} \times \mathbf{n}_f}{\|\mathbf{t} \times \mathbf{n}_f\|_2}, \qquad \mathbf{p}_b = \begin{bmatrix} \mathbf{n}_b \\ -\mathbf{n}_b^\top \mathbf{u} \end{bmatrix}. \tag{26}$$

With the reported boundary weight of 10, this construction contributes $10\,\mathbf{p}_b\mathbf{p}_b^\top$ to the endpoint quadrics. Each endpoint contribution to the error is $10[\mathbf{n}_b^\top(\mathbf{x} - \mathbf{u})]^2$. Together with the incident surface plane, it discourages movement away from the boundary line while permitting movement along it. This soft geometric penalty is combined with topology and orientation checks when accepting collapses.

Admissibility and simultaneous edits. The full simplicial link condition in Eq. (9) includes both vertices and edges in the links. Boundary, merged-fan, degeneracy, and orientation tests further screen candidate changes for local validity. Within one update, accepted collapses have disjoint affected one-rings, which prevents two selected collapses from simultaneously modifying the same local connectivity.

Refinement and a conforming edge subdivision. The refinement score multiplies a vertex-position gradient magnitude by its camera-space depth and averages across the contributing views. This score prioritizes vertices whose positions remain sensitive to the image-space objective. For a selected edge (i, j), the new midpoint and its initial appearance attributes follow the interpolation formulas in the main paper. For the subdivision to remain conforming, the same midpoint must be shared by all incident faces of that edge: an interior edge has two such faces and a boundary edge has one. Each incident triangle is replaced by two triangles with consistent winding. Sharing the midpoint across incident faces avoids T-junctions in the refined neighborhood.

Parameter interpolation and optimizer-state interpolation. The attribute and moment-transfer rules serve different purposes. Collapsed appearance attributes use opacity-weighted averaging, whereas Adam moments use the incident-area weights in Eq. (16). Writing $\eta = A_u/(A_u + A_v)$ for positive total incident area gives $m_{\text{new}} = \eta m_u + (1 - \eta)m_v$ and $v_{\text{new}} = \eta v_u + (1 - \eta)v_v$. The latter remains elementwise nonnegative because it is a convex combination of second-moment estimates. For splitting, the two perturbed first moments in Eq. (17) satisfy

$$\frac{m_k^+ + m_{k'}}{2} = m_k, \qquad v_k^+ = v_{k'} = v_k. \tag{27}$$

The opposite perturbations therefore preserve the pair's mean first moment while breaking identical inherited states. These transfers initialize the updated parameterization from the accumulated optimization history.

A.4 Residual Texture Coordinates, Filtering, and Capacity

A continuous transition to appearance refinement. The rendered color combines the SH-based appearance with a spatial residual. At initialization, $T_f = 0$ for every face, and average pooling and linear texture interpolation both map zero to zero. Consequently, $C^{\lambda}_{\text{res}} = 0$ at every mip level, so adding the residual branch initially preserves the base rendering. During refinement, SH coefficients and textures are optimized jointly. The residual models spatial variation in local face coordinates, while the SH coefficients provide view-dependent appearance.

Triangular support within a square texture. The barycentric coordinates $(u, v) = (w_j, w_k)$ belong to $\mathcal{D} = \{(u, v) : u \geq 0, v \geq 0, u + v \leq 1\}$. The mirror extension can be written explicitly as

$$\mu(u, v) = \begin{cases} (u, v), & u + v \leq 1, \\ (1 - v, 1 - u), & u + v > 1. \end{cases} \tag{28}$$

Reflection fixes the diagonal $u + v = 1$ and maps the complementary half of the unit square back into $\mathcal{D}$. It provides values outside the triangle for square-grid interpolation and filtering within each face.

Screen-space footprint and coordinate units. The LOD estimate is most explicit when its derivatives are expressed in texel units. For an $R \times R$ base texture and normalized (u, v), define

$$\begin{aligned} \rho_x &= R\sqrt{(\partial u/\partial x)^2 + (\partial v/\partial x)^2}, \\ \rho_y &= R\sqrt{(\partial u/\partial y)^2 + (\partial v/\partial y)^2}, \\ \lambda_{\text{tex}} &= \log_2(\max(\rho_x, \rho_y) + \epsilon)\,. \end{aligned} \tag{29}$$

Here x, y are pixel coordinates and $\rho = \max(\rho_x, \rho_y)$ measures texels covered per pixel. The factor R is needed for normalized UVs; it is already absorbed when derivatives are taken in texel coordinates. Equation (29) makes the normalized-coordinate formulation explicit, following the texel-space footprint convention in standard texture filtering. For mip levels $0, \ldots, L$, valid level selection uses

$$\hat{\lambda} = \min\{L, \max\{0, \lambda_{\text{tex}}\}\}, \quad l_0 = \lfloor \hat{\lambda} \rfloor, \quad l_1 = \min(l_0 + 1, L), \quad \delta = \hat{\lambda} - l_0. \tag{30}$$

The filtered residual is then

$$C_{\text{res}} = (1 - \delta)\,\text{Bilinear}(T_f^{l_0}, u, v) + \delta\,\text{Bilinear}(T_f^{l_1}, u, v). \tag{31}$$

This form makes the finest- and coarsest-level boundary cases explicit. Because coarse levels are produced by average pooling, their gradients propagate to the underlying fine-level texture. Isotropic mip filtering reduces high-frequency texture content under minification.

Appearance capacity and storage scaling. For F faces, the dense base-level RGB grids contain $3FR^2$ scalar entries. When $R = 2^L$, storing every mip level requires

$$3F \sum_{l=0}^{L} (R/2^l)^2 = 3F \frac{4R^2 - 1}{3} < 4FR^2 \tag{32}$$

scalar entries. For $R = 8$, a full pyramid has $64 + 16 + 4 + 1 = 85$ RGB texels per face, compared with 64 at its base level. Memory use also depends on precision, gradients, optimizer states, and the storage of intermediate mip levels. Increasing R from 4 to 8 quadruples the base-grid capacity at a fixed face count. Conversely, reducing the number of faces also reduces the total number of face textures, even when each remaining face has the same resolution.

A.5 Mesh Export and Rendering Efficiency

Both PLY and GLB export first aggregate SH colors, including higher-order contributions, over visible training views and incorporate residual textures to produce static baked appearance. Both formats preserve the face count.

Table 8: **Geometric reconstruction on Tanks & Temples.** F-scores (↑) on six scenes and their arithmetic mean. The mean for T^2-Splat is computed from unrounded per-scene scores.

	Foreground only		Full scene extraction				
Scene	2DGS	Gaussian Wrapping	GOF	RaDe-GS	MiLo	Gaussian Wrapping	**T^2-Splat**
Barn	0.51	0.67	0.49	0.47	0.52	0.67	0.47
Caterpillar	0.31	0.50	0.29	0.30	0.33	0.50	0.26
Courthouse	0.10	0.16	0.11	0.11	0.12	0.18	0.26
Ignatius	0.61	0.78	0.63	0.66	0.74	0.78	0.66
Meetingroom	0.19	0.41	0.24	0.25	0.23	0.41	0.21
Truck	0.54	0.63	0.50	0.57	0.56	0.63	0.50
Mean	0.38	0.53	0.38	0.39	0.42	0.53	0.40

Table 9: Geometry and topology statistics of the final meshes on the nine scenes of Mip-NeRF 360. Triangle shape quality is reported as the equally weighted average of the per-scene median values.

	Average faces (M)	Non-manifold edges (%) ↓	Non-manifold vertices (%) ↓	Triangle shape quality ↑	Faces in largest component (%)
A: Baseline	5.308	17.84	65.06	0.5796	52.94
B: Safe RDT	3.049	**0**	**0**	0.5918	44.30
C: Safe RDT + DTE	**2.468**	**0**	**0**	**0.6203**	**54.41**

Vertex-colored PLY. Residual textures are sampled and fitted to three corner-color residuals per face. The corner residuals are combined at shared vertices using face-area weights and added to the baked SH colors. The file stores positions, RGB vertex colors, and face indices, preserving the geometry and connectivity. The baked vertex colors consolidate the SH and residual texture contributions into a single vertex-color representation.

Texture-atlas GLB. Interpolated SH colors and residual textures are combined into per-face RGB textures and packed into an atlas. The GLB contains geometry, UV coordinates, embedded PNG textures, and materials. The face count is unchanged, while vertices are expanded into three corners per face to accommodate independent UV coordinates. This retains more intra-face appearance capacity than vertex color fitting, at the cost of additional storage.

OpenGL rendering efficiency. Table 3 evaluates the baked vertex-colored PLY on Garden at 1920×1080 on a MacBook Air M3, reporting 236 FPS and a file size of 62 MB. Relative to MeshSplatting under that OpenGL setup, the displayed values imply 1.87 times the frame rate and a 59.2% reduction in file size.

A.6 Additional Results and Analysis

A.6.1 Surface reconstruction and topology

Geometric reconstruction on Tanks & Temples. Table 8 reports F-scores on six scenes using scale-corrected virtual scanning. T^2-Splat achieves an arithmetic mean of 0.40, computed from unrounded per-scene values, and the highest Courthouse score of 0.26 among the listed methods. In the full-scene columns, MiLo and Gaussian Wrapping obtain means of 0.42 and 0.53, respectively. The table groups foreground-only and full-scene extraction results separately. These surface measurements complement the NVS evaluation.

Complementary evidence from DTU. Table 2 reports lower Chamfer distance than MeshSplatting on 14 of the 15 DTU scans; scan 97 is the exception (1.18 versus 1.15). The reported mean decreases from 0.89 to 0.68, a reduction of approximately 23.6%. T^2-Splat achieves the lowest reported error on scans 63, 83, and 105. Its mean is tied with RaDe-GS and MiLo, while Gaussian Wrapping has a lower mean of 0.49.

Compactness and local mesh quality. Table 9 reports geometry statistics on nine Mip-NeRF 360 scenes. Safe RDT reduces the average face count from 5.308M to 3.049M (42.6%). DTE further reduces it to 2.468M (19.1% relative to Safe RDT), for an overall reduction of 53.5% relative to the baseline. Both constrained configurations have zero measured non-manifold edge and vertex ratios in these evaluations. The reported triangle-quality statistic is the equally weighted average of the per-scene median values. DTE increases this statistic from 0.5918 to 0.6203, an absolute gain of 0.0285.

Component connectivity. Safe RDT decreases the fraction of faces in the largest component from 52.94% to 44.30%, while reducing the local non-manifold ratios to zero. Adding DTE raises this fraction to 54.41%, a gain of 10.11 percentage points over Safe RDT. Together with the reduced face count and improved triangle quality, this shows how DTE changes both mesh compactness and the distribution of faces across connected components.

A.6.2 Component contributions and resource trade-offs

Controlled comparisons in the ablation table. Table 10 summarizes selected differences from Table 5. Comparing rows with the same texture resolution shows the effect of DTE, while rows with the same geometry configuration show the effect of appearance refinement. The appearance-refinement stage jointly optimizes SH coefficients and textures.

Table 10: Differences derived from the displayed values in Table 5. Positive PSNR/SSIM changes and negative LPIPS changes indicate improvements. Time differences refer to reported total training time.

Comparison	ΔPSNR (dB)	ΔLPIPS	ΔSSIM	Δtime (min)
(b)→(c): add DTE	+0.31	-0.02	+0.016	+14
(e)→(g): add DTE with 8^2 textures	+0.18	-0.01	+0.010	+2
(c)→(f): add 4^2 refinement	+0.31	-0.03	+0.019	+15
(f)→(g): increase 4^2 to 8^2	+0.06	-0.03	+0.004	+11
(a)→(g): complete configuration	+0.83	-0.07	+0.045	+29

With DTE enabled, increasing texture resolution from 4×4 to 8×8 quadruples base texture capacity, improves average PSNR by 0.06 dB, and reduces LPIPS by 0.03. Total training time rises from 72 to 83 minutes, illustrating the quality–cost trade-off of higher texture resolution. Configurations (e) and (g) differ by 2 minutes in total training time and also use different face counts.

Mesh budgets on Garden. At the 50% face-budget setting, PSNR decreases by 0.30 dB, LPIPS increases by 0.02, and SSIM decreases by 0.017 relative to the full model. The resulting 1.4M-face mesh still improves over the listed MeshSplatting result by 0.68 dB PSNR, 0.05 LPIPS, and 0.039 SSIM while using approximately 80.8% fewer faces than its 7.3M-face mesh. At the 10% setting, the corresponding changes relative to the full model are -1.75 dB PSNR, $+0.15$ LPIPS, and -0.121 SSIM. These settings provide a tunable balance between rendering quality and mesh size.

A.6.3 Per-scene novel view synthesis

Tables 11, 12, and 13 report per-scene measurements for the two Tanks & Temples NVS scenes and all nine Mip-NeRF 360 scenes. The two-scene NVS evaluation uses a different protocol and scene set from the six-scene geometric evaluation above; Truck appears in both. Among the outdoor Mip-NeRF 360 scenes, Garden has the lowest LPIPS (0.148) and highest SSIM (0.818), whereas Flowers has the lowest PSNR (19.81 dB) and SSIM (0.546). Indoor PSNR values range from 27.01 dB on Counter to 29.36 dB on Bonsai. For Tanks & Temples, Truck exceeds Train by 3.59 dB PSNR and 0.083 SSIM, and has 0.092 lower LPIPS.

Qualitative comparisons. Figure 4 shows Garden, Shelf, Truck, and Train, including enlarged image regions. These examples allow visual comparison of local texture patterns and edge detail against ground truth, MeshSplatting, and MiLo. Shelf is an additional STRinGS-360 example and is not part of the nine-scene Mip-NeRF 360 or two-scene Tanks & Temples NVS averages.

Table 11: Per scene LPIPS, PSNR, and SSIM scores for the Truck and Train scenes of the T&T dataset.

	Truck	Train
PSNR↑	22.91	19.32
LPIPS↓	0.167	0.259
SSIM↑	0.836	0.753

Table 12: Per-scene PSNR, LPIPS, and SSIM scores for outdoor MipNeRF360 scenes.

	Bicycle	Flowers	Garden	Stump	Treehill
PSNR↑	23.96	19.81	25.68	25.36	21.01
LPIPS↓	0.247	0.336	0.148	0.252	0.327
SSIM↑	0.709	0.546	0.818	0.723	0.595

A.7 Limitations and Future Directions

Surface smoothness and geometric completeness. The reconstructed mesh still exhibits local surface undulations and normal discontinuities, while the recovery of thin structures and sparsely observed regions also depends on the initial geometric support. Future work could jointly train the explicit mesh with a signed distance field (SDF), combining continuous geometric constraints with mesh rendering and topology optimization. Surface-position and normal consistency could couple the two representations. Together with multi-view observations and moderate smoothing regularization, this approach could improve surface smoothness and geometric coherence while preserving sharp edges and thin structures.

Local topology updates and global geometry. Connectivity checks evaluate the neighborhoods affected by an edit, whereas surface self-intersections depend on the spatial arrangement of faces that need not share a local neighborhood. Continuous vertex optimization can also fold a locally valid mesh. Efficient collision queries and displacement constraints could therefore complement the topology checks during both editing and subsequent optimization. The current discrete editing procedure additionally involves edge extraction, candidate ranking, and transfers between CPU and GPU memory. Incremental adjacency maintenance, local candidate updates, and parallel optimizer-state transfer offer directions for reducing this overhead on larger scenes.

Geometry and texture resource allocation. Dense per-face RGB textures contain $3FR^2$ scalar parameters for F faces and resolution R, with additional storage for gradients, optimizer states, and mip levels. A uniform resolution assigns the same capacity to faces with very different projected sizes and appearance complexity. Reducing the face count also reduces the number of texture grids, so larger retained triangles must represent more surface detail within each grid. Joint allocation of face and texel budgets could account for projected area, image residuals, and view coverage. Adaptive texture resolution, shared texture bases, and quantization could further improve storage efficiency under a specified quality budget.

Texture continuity and oblique views. Independent face textures can assign different colors to the same shared edge. The mirror extension provides samples outside each triangular domain, but does not coordinate the values of neighboring faces. Cross-face consistency terms, shared edge samples, or jointly optimized atlas boundaries could improve continuity. Filtering presents a related challenge: isotropic mip levels use a single scale for footprints that can become strongly elongated at grazing angles. Differentiable anisotropic filtering based on the projected texture footprint could better preserve detail in the short-footprint direction while suppressing aliasing along the long direction.

Table 13: Per-scene PSNR, LPIPS, and SSIM scores for indoor MipNeRF360 scenes.

	Room	Counter	Kitchen	Bonsai
PSNR↑	28.79	27.01	28.45	29.36
LPIPS↓	0.224	0.214	0.150	0.214
SSIM↑	0.904	0.885	0.909	0.924

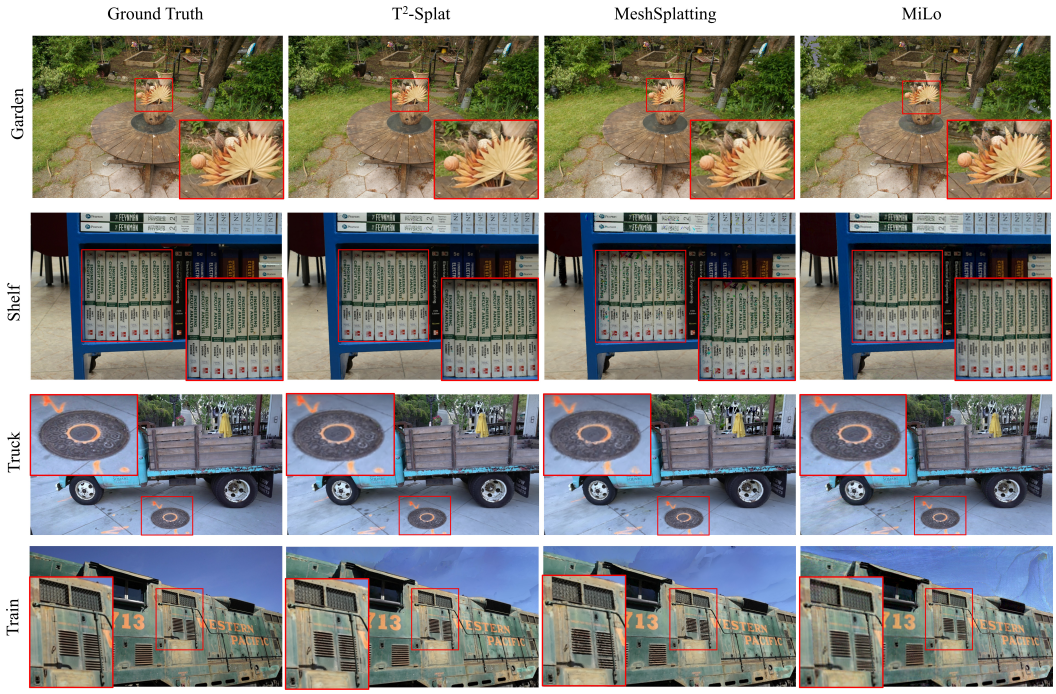


Figure 4: Additional qualitative comparisons on Garden, Shelf, Truck, and Train. The columns show ground truth, T^2-Splat, MeshSplatting, and MiLo. The highlighted regions provide close views of local texture and edge detail. Shelf is an additional STRinGS-360 example, outside the NVS averages reported in the per-scene tables.

Complex appearance and lighting variation. Vertex SH coefficients represent directional appearance, while residual textures describe spatial variation in fixed face coordinates. Sharp specular highlights combine both types of variation and can demand more capacity than this decomposition provides. Transparency and refraction additionally involve layered visibility and light transport through surfaces. Direction-conditioned residuals, local material models, and layered surface representations are possible extensions for these effects. The learned appearance also incorporates the illumination of the capture environment. Separating material properties from lighting would enable editing and relighting beyond the captured conditions.

Export fidelity and deployment. Static color baking compresses the trained appearance for conventional mesh rendering. It removes directional variation, and fitting a face texture to three corner colors can smooth spatial detail within the face. Texture-atlas export retains more of this variation but increases texture and UV storage. Future export methods could optimize the representation directly for the target renderer and a chosen asset-size or frame-time budget. Combining mesh levels of detail with texture resolution selection and atlas packing could support scalable deployment; a lightweight directional shader could retain part of the SH appearance when the rendering environment permits it.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We introduce T^2-Splat, a unified differentiable rendering framework that bridges the gap between high-fidelity 3D Gaussian Splatting and industry-standard polygon pipelines. By integrating dynamic topology evolution with scale-adaptive texturing, our method achieves superior rendering quality and surface reconstruction with a fraction of the geometric complexity of existing baselines. Please see our abstract and introduction for further details.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our method still struggles with the geometric crumpling and transparent objects like most mesh-based reconstruction methods. Please see the appendix for further details.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In the method and experiment sections, we provide a detailed discussion to demonstrate each result.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have disclosed the core technical information, implementation details, and evaluation methods required to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will release the code after acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the core experimental setup, hardware environment, and training hyperparameters, and explicitly indicated that more comprehensive details are included in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experimental result for rendering and reconstruction tasks from dense views are very stable, the standard deviation is negligible, so we follow the previous work's practice and not report the error bar.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide sufficient information on the computer resources, including the type of compute workers GPU.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have carefully reviewed the NeurIPS Code of Ethics and are committed to strictly adhering to its guidelines.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: Neural rendering and 3D reconstruction are crucial for downstream applications in VR, AR, and game engines. Because our framework successfully outputs lightweight, high-fidelity meshes that directly integrate into traditional graphics pipelines, we believe it has the strong potential to advance progress in this domain and support real-world deployments.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: There are no such risks in the rendering and reconstruction tasks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: In the Experiments section, we cite all used code package and public datasets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: We will release the code after acceptance.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve any crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We only use the LLM for writing, editing, and formatting purposes, which does not affect the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.