AUTOMATIC NEURAL BAKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Baking precomputes an object's appearance into a texture on a mesh, but baked formats cannot represent sharp view-dependent reflection. Neural and splat representations can, yet they place appearance in the space *around* the object rather than on its surface, so there is nothing to bake. We identify two independent obstructions to the natural fix, binding splats to the faces of a known mesh, and prove both are fundamental rather than incidental. First, coplanar primitives admit no compositing order that is both view-stable and geometrically meaningful, and no cheap, per-primitive perturbation of the depth, such as a normal offset, can repair this. Second, reflection about a fixed normal is a rotation of the view sphere, so any per-primitive view-dependent color model is band-limited at a fragment regardless of how many primitives are used: adding primitives adds no angular bandwidth. We resolve both with Automatic Neural Baking, a construction that works rather than the only one that could. An *on-manifold kernel field*, 2D Gaussian disks anchored to mesh faces and blended by an order-free partition of unity, removes the ordering obstruction; factoring view dependence out of the elements into a shared environment map and a FastKAN decoder removes the locality bound. The result composes directly with the traditional rendering pipeline, baking splat-quality, real-time appearance onto a mesh.

1 INTRODUCTION

Interactive applications such as games, virtual reality, and other online rendering, demand photorealistic imagery in real time (typically $\geq$ 60 FPS). The standard baking-and-rasterization pipeline meets this by precomputing appearance offline into textures over the mesh, which a fragment shader then samples cheaply at run time (Akenine-Moller et al., 2019).

Automating this stage, however, remains hard for the shiny, view-dependent materials that matter most: the usual baked formats, for instance, lightmaps or low-order Spherical Harmonic (SH) probes, store only smooth, view-independent lighting and miss angle-dependent highlights (Ng et al., 2003; Kuznetsov et al., 2022; Huang et al., 2024b). Baking also presupposes a global UV atlas, whose automatic construction remains fragmented and largely manual (Lévy et al., 2002; Poranne et al., 2017; Li et al., 2018; Srinivasan et al., 2024). Per-face texturing such as Ptex (Burley & Lacewell, 2008) removes the UV atlas, but as a static texture it still cannot capture high-frequency, view-dependent reflection. The seminal automatic neural baking method is Deferred Neural Rendering (DNR) (Thies et al., 2019), which bakes learned neural textures onto a mesh and decodes them at render time. However, DNR shares the same weakness as classical baking and cannot reproduce high frequency view-dependent color. Our method is a direct, modern successor to it: the mesh is an input, an artist-authored asset as in standard baking workflows, and unlike DNR we also capture the sharp, view-dependent appearance that baking has always lacked.

Such appearance has instead been the domain of neural and splat representations. Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020) and 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) have revolutionized novel view synthesis, reaching a photorealism that classical baking cannot match by treating the scene as a continuous optimization problem. But they place appearance in the *ambient space*: a continuous field over $\mathbb{R}^3$ for NeRF, a mixture of anisotropic kernels for 3DGS. The baking pipeline instead treats an object as a *surface*, a two-dimensional manifold $\mathcal{M} \subset \mathbb{R}^3$ carrying appearance as a field on $\mathcal{M}$ itself. Because the ambient field for NeRF and 3DGS has no 2D domain, there is nothing to bake. In practice this means splat methods produce renderers, not assets: their

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

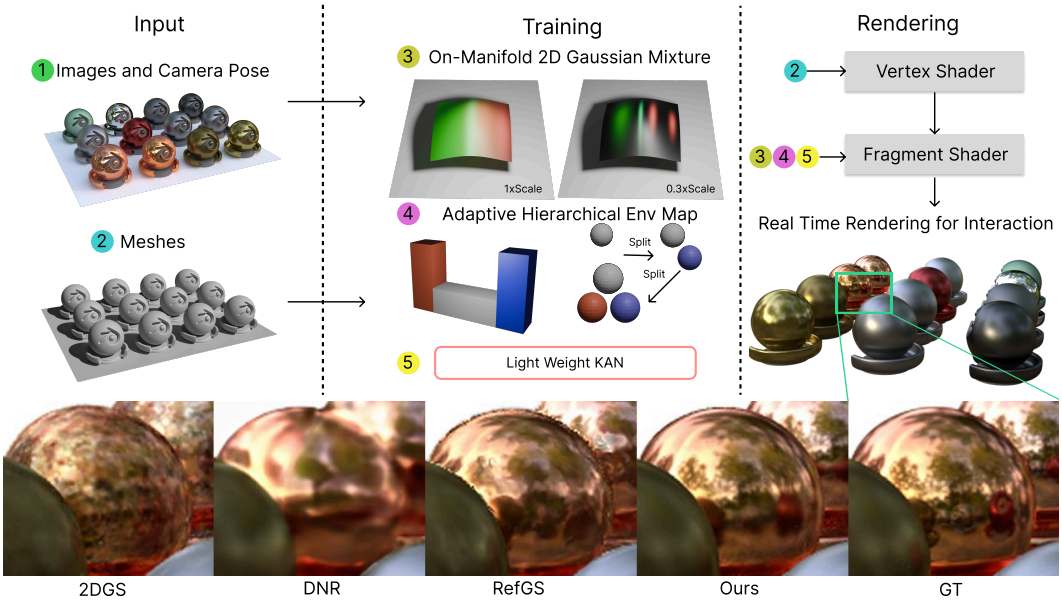

Figure 1: **Automatic Neural Baking bakes highly reflective, view-dependent appearance directly onto a mesh, rendering it in real time ($\geq$ 60 FPS) given the same input mesh that flexible-mesh methods must instead reconstruct.** *Comparison*: on Shiny-Blender, our method reproduces sharp, high-frequency specular reflections that 2DGS* (Huang et al., 2024a) and DNR (Thies et al., 2019) smear and that RefGS (Yao et al., 2025) recovers less crisply, while running natively in an nvdiffrast rasterizer rather than requiring depth-sorted splat rendering. *Pipeline*: from posed reference images 1 and the corresponding triangle mesh 2, we optimize three interchangeable, rasterizer-native modules—an on-manifold 2D Gaussian mixture storing diffuse albedo and reflectivity 3, an adaptive, hierarchical environment map capturing self-reflection 4, and a lightweight KAN that evaluates the reflection equation for far-field specular 5—all evaluated per-fragment in a form a standard fragment shader can execute 2 3 4 5.

quality cannot enter a pipeline that assumes a surface. This paper supplies that missing step, not a replacement for the pipeline downstream of it.

Attempts to close this gap do not change the domain. Extraction methods recover a mesh after the fact (Guédon & Lepetit, 2024; Dai et al., 2024) but only approximate a surface the representation never carried, and the primitives themselves remain free in $\mathbb{R}^3$. 2DGS (Huang et al., 2024a) flattens each ellipsoid into a disk, collapsing the kernel across its normal direction; this removes thickness, but a sum of such disks scattered through space is still an ambient-space distribution. The natural remaining step is to bind the primitives directly to the faces of a known mesh. This is where the difficulty becomes concrete.

Once primitives are strictly coplanar, their depth keys tie and alpha compositing has no well-defined order. The standard remedies fail in instructive ways. Offsetting disk i along the face normal by ε_i separates two disks by $(\varepsilon_i - \varepsilon_j)(\mathbf{n} \cdot \mathbf{v})$, where $\mathbf{n}$ is the face normal and $\mathbf{v}$ the viewing direction. With small offsets, pairs still flip at views reaching high elevations, and flips are densest where the surface faces the camera; larger offsets push the flips toward grazing angles but never remove them, so gradients from different views still disagree about which primitive owns a fragment. Large offsets also let the optimizer reintroduce extent along $\mathbf{n}$, reinstating the ambient drift the binding was meant to remove and degrading the surface it is bound to. Breaking ties by a fixed primitive index is view-stable but imposes an occlusion relation uncorrelated with geometry. The obstruction is not depth but *order-dependence itself*: on a single opaque surface no ordering is both view-stable and geometrically meaningful. We prove this formally as Theorem B in App. A.1: no fix that keeps sort-based compositing with a static per-primitive perturbation of the depth key, a normal offset being the canonical example, can avoid it. Every finite offset still flips some pair at a visible view, and suppressing all flips to within a grazing angle requires an offset that diverges as the angle shrinks, turning the surface into a volume. We confirm this experimentally in Sec. 4.1.

We therefore drop ordering altogether, and bake appearance as a field built from kernels sited on the surface itself: two-dimensional Gaussian disks anchored to mesh faces in barycentric coordinates, blended at each fragment by a normalized partition of unity over the K nearest disks. A convex combination is invariant to the order of its terms, so the global $O(N \log N)$ sort becomes an $O(K)$ local gather with no degeneracy to avoid. On this field we place the view-dependent term, and the construction turns out to carry more than the problem that motivated it.

Per-primitive view-dependent color would seem a natural way to add reflection to this field, but a second, independent obstruction rules it out. Reflection about a fixed normal is a rotation of the view sphere, so any low-order, per-primitive color model occupies a fixed-dimensional function space at every fragment, and adding primitives adds no angular bandwidth to escape it (Theorem A1, App. A.1). Only a shared, high-capacity representation read by a per-fragment decoder can exceed this floor, which is why the view-dependent term above is not one more per-disk attribute but the environment map and KAN decoder of Secs. 3.3–3.4.

Our central contribution is the representation; the remaining items are properties it makes possible.

- **On-manifold kernel field.** We answer both obstructions with one construction: a field built from kernels sited on the surface, each element carrying its own value, blended at a fragment by a smooth partition of unity over its K nearest neighbours. Because the kernels are placed in barycentric coordinates, the field needs no global UV atlas; because a convex blend is invariant to the order of its terms, the coplanar degeneracy cannot arise. This is a resolution, not the resolution: a fixed-index order also satisfies Theorem B, and whether it can be paired with the same shared decoder to match this representation is a question we leave open (App. A.1). To our knowledge this is the first splat representation developed as an element-wise texture: a field whose elements are individually addressable, differentiable, and mergeable, and whose domain is the surface itself, rather than a distribution in ambient $\mathbb{R}^3$. Mergeability matters specifically for baking: baked assets have long relied on precomputed levels of detail, such as mipmaps, to trade quality for size, and because our elements are ordinary splats, tools built for the Gaussian-splatting community compact them directly, giving an automatic analogue of mipmap generation (Sec. 4.5).
- **Sharp view-dependent appearance, baked.** A FastKAN decoder over an adaptive hierarchical environment map captures high-frequency specularity, and near-convex decomposition reduces self-reflection to a local lookup rather than a secondary ray. On Shiny-Blender we reach 38.04 dB PSNR, the best reported result given the same geometry.
- **Serves as a texture.** By construction, our field is read like a texture: a local, per-fragment lookup with no UV atlas and no depth sort, though the lookup itself runs a small decoder rather than a single sample. We realize it in nvdiffrast and render both datasets at over 60 FPS, while reaching the novel-view-synthesis quality of splat representations.

2 Related Work

2.1 Baking Appearance onto Explicit Geometry

Baking precomputes an object's appearance into reusable textures (Akenine-Moller et al., 2019). Deferred Neural Rendering generalized this recipe by storing learned feature textures on the mesh and decoding them in screen space (Thies et al., 2019). With the advent of NeRF (Mildenhall et al., 2020), baking became a conversion problem: trained fields are distilled into sparse feature grids (Hedman et al., 2021), textured polygon soups (Chen et al., 2023), watertight meshes (Yariv et al., 2023), or deformable avatars (Duan et al., 2023), as well as differentiable inverse rendering recovers meshes, materials, and lighting directly from images (Munkberg et al., 2022), and neural textures can be synthesized over curved surfaces (Huang et al., 2024b). These methods deliver genuine assets, but their view-dependent capacity is an afterthought: low-order SH lobes or small per-pixel decoders biased toward diffuse content. Our pipeline retains the deferred, texture-like structure of this line while devoting explicit model capacity to sharp specularity. In addition, methods above presuppose a global UV atlas, whose seam-cutting and packing remain tedious to author and non-differentiable to optimize; by anchoring appearance in per-face barycentric coordinates, our formulation sidesteps this requirement entirely. Per-face texturing such as Ptex (Burley & Lacewell,

2008) likewise removes the atlas, but its static per-face textures cannot represent the high-frequency, view-dependent reflection that continuous splat representations later captured.

2.2 Modeling View-Dependent Appearance

A parallel line of work asks how to reconstruct sharp specular transport in the first place. Ref-NeRF reparameterizes outgoing radiance about the reflected view direction, substantially improving specular fidelity within a volumetric field (Verbin et al., 2022). In the splatting regime, recent methods evaluate reflections in screen space against a learned environment map: GaussianShader attaches simplified shading functions to each splat (Jiang et al., 2024), 3DGS-DR defers reflection evaluation to a per-pixel normal buffer (Ye et al., 2024), RefGS incorporates physically based deferred shading (Yao et al., 2025), and Spec-Gaussian replaces spherical harmonics with anisotropic spherical Gaussians (Yang et al., 2024).

Whether it is an evaluated BRDF (Cook & Torrance, 1982) against a prefiltered environment map, as in classical real-time physically based shading (Karis, 2013; Akenine-Moller et al., 2019), or learned as above, environment-map reflection shares one limitation: it assumes all incident light arrives from infinitely far away. Reducing a reflection to a single directional lookup, it cannot represent near-field self-reflection or self-occlusion, the light an object casts back onto itself. Recovering these effects generally requires tracing secondary rays (Yao et al., 2025; Moenne-Loccoz et al., 2024), which reintroduces the very queries that rasterization avoids. We instead handle self-reflection geometrically: because a convex part can neither occlude nor reflect itself, we split an object into near-convex parts via convex decomposition (Wei et al., 2022), each carrying its own environment map, grown as an adaptive hierarchy that isolates high-error regions, so that self-reflection reduces to local lookups rather than recursive rays.

These methods solve specular *reconstruction*, yet all of them operate on free-form primitives in ambient $\mathbb{R}^3$: they expose no solid surface, retain the global depth sort, and produce nothing that can be baked onto a given mesh. Naively binding them to a fixed surface only forces them co-planar and destabilizes the sort; our surface-bound 2D formulation sidesteps this by construction, requiring no sorting at all.

2.3 Attaching Radiance Fields to Surfaces

A third line pushes radiance distributions toward surfaces. Flattened kernels such as 2D Gaussian disks (Huang et al., 2024a) and Gaussian surfels (Dai et al., 2024) collapse one axis of the kernel to improve geometric accuracy, yet the primitives still float freely in $\mathbb{R}^3$. Mesh-scaffolded methods go further: SuGaR regularizes splats toward an extracted mesh (Guédon & Lepetit, 2024), GaMeS parameterizes Gaussians on triangle faces to enable editing (Waczyńska et al., 2024), SplattingAvatar embeds splats in a deformable template for animation (Shao et al., 2024), and Texture-GS learns a UV mapping so that a 2D texture colors the splats (Xu et al., 2024). In every case, however, the mesh merely *drives* the kernels: the rendered primitives remain three-dimensional, ambient-space objects composited by depth sorting, and the domain of the appearance field never becomes the surface itself. Yang et al. (2022) bind learnable codes on the vertices, but never allow the code representation to float inside the 2D manifold. The original EWA surface splatting (Zwicker et al., 2001) is, moreover, not differentiable. In contrast, we define a strictly 2D, sort-free Gaussian mixture expressed in the barycentric coordinates on the faces that is differentiable and free inside the 2D manifold. In this way the mesh is not scaffolding but the domain, and it remains the authoritative solid geometry for collision, physics, and animation.

3 Method

3.1 Overview

Given a triangle mesh M characterized by vertices V and faces F, along with a set of posed reference images (I), we bake the object's appearance into a field that a rasterizer can evaluate per fragment in real time. The mesh is fixed; only the appearance on it is optimized.

We bake appearance with three modules and composite their outputs at each fragment (Figure 2). An on-manifold 2D Gaussian mixture (Sec. 3.2) carries the view-independent fields; an adaptive

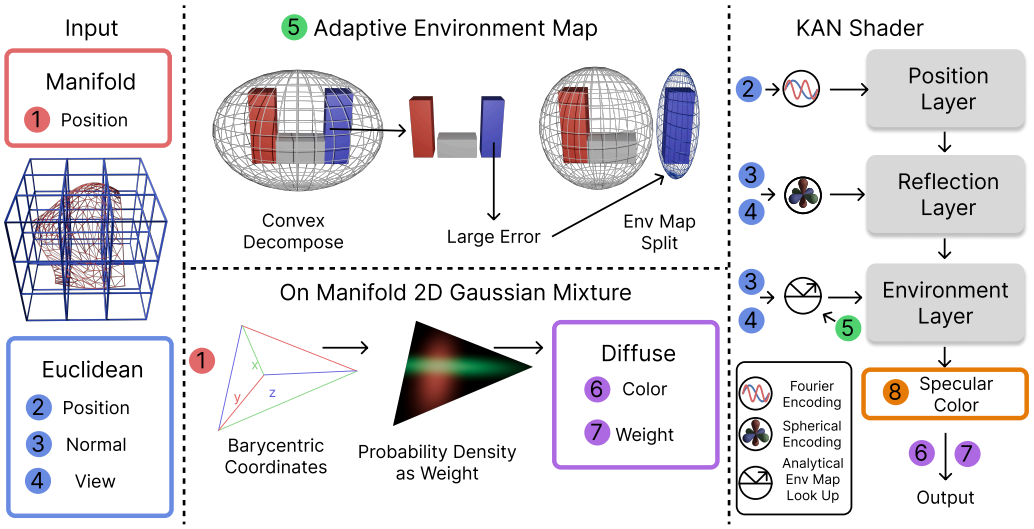

Figure 2: **Pipeline overview.** *Input:* two position types—the on-manifold surface position ❶ (barycentric coordinates on a mesh face) and the Euclidean position ❷, plus the surface normal ❸ and view direction ❹. *On-manifold 2D Gaussian mixture* (Sec. 3.2): from the barycentric position ❶ we gather nearby surface Gaussians by their probability density (a partition-of-unity weight), giving the diffuse color ❻ and per-point reflectivity ❼. *Adaptive environment map* ❺ (Sec. 3.3): the object is convex-decomposed into near-convex parts, and the highest-error region is recursively split off into its own map. *KAN shader* (Sec. 3.4): a FastKAN decoder consumes the Fourier-encoded position ❷ (Position Layer), the spherically-encoded normal ❸ and view ❹ (Reflection Layer), and an analytical lookup into the environment map ❺ (Environment Layer), producing the specular color ❽. *Output:* the final color composites the specular ❽ with the diffuse color ❻ gated by the reflectivity ❼, i.e. Eq. (1).

environment map (Sec. 3.3) supplies the incident radiance, including the light the object casts on itself; and a compact FastKAN decoder (Sec. 3.4) turns a lookup into that map into a view-dependent reflection. At a visible surface point $\mathbf{x}$ seen along view direction $\mathbf{v}$, the three combine as

$$c(\mathbf{x}, \mathbf{v}) = \mathrm{sigmoid}\Big(\mathrm{logit}\,\big(a(\mathbf{x})\big) + m(\mathbf{x})\, r(\mathbf{x}, \mathbf{v}) \Big), \tag{1}$$

where the albedo $a(\mathbf{x}) \in [0,1]^3$ and the reflectivity $m(\mathbf{x}) \in [0,1]$ are read from the mixture (m=0 matte, m=1 mirror) and the residual $r(\mathbf{x}, \mathbf{v})$ is emitted by the decoder. Compositing in logit space lets the reflection act as an additive residual on the diffuse base while keeping the result in $[0,1]$ without clamping; m gates where it applies.

Training. All modules are fit jointly by minimizing a photometric loss over the training views,

$$\mathcal{L}_{\text{app}} = (1-\lambda)\,\mathcal{L}_1\big(I, I^{\text{gt}}\big) + \lambda\,\mathcal{L}_{\text{D-SSIM}}\big(I, I^{\text{gt}}\big), \tag{2}$$

with no regularizers on the disks. We utilize the Adam Optimizer (Kingma & Ba, 2015), with 30K iterations.

3.2 2D Gaussian Mixture on the Manifold

The albedo and reflectivity fields are stored by a mixture of N two-dimensional Gaussian "disks" on the mesh surface (N=10^6 in our experiments). N is fixed at initialization and never changes during training: there is no densification, pruning, or opacity culling.

On-manifold anchoring. Each disk i is assigned to a single triangle f_i and a barycentric coordinate $\mathbf{b}_i = (b_{i,0}, b_{i,1}, b_{i,2})$ with $b_{i,k} \geq 0$ and $\sum_k b_{i,k} = 1$. Its center is *reconstructed* from the live vertices $\mathbf{p}_0(f_i), \mathbf{p}_1(f_i), \mathbf{p}_2(f_i)$ of that triangle,

$$\mathbf{x}_i = b_{i,0}\,\mathbf{p}_0(f_i) + b_{i,1}\,\mathbf{p}_1(f_i) + b_{i,2}\,\mathbf{p}_2(f_i), \tag{3}$$

so $\mathbf{x}_i$ is by construction a convex combination of the face's vertices and lies exactly on the triangle. The face therefore supplies the disk's frame as well as its position: its extent lives in the in-plane axes $(\mathbf{t}_1, \mathbf{t}_2)$ of f_i, with an in-plane rotation θ_i about the face normal $\mathbf{n}$ and an anisotropic in-plane scale $(s_{i,1}, s_{i,2})$. We give the disk *no extent along* $\mathbf{n}$, so it is a strictly two-dimensional element rather than a three-dimensional one with a constrained center.

Per-disk attributes. Each disk carries two view-independent attributes: an albedo $a_i \in [0,1]^3$ and a scalar reflectivity $m_i \in [0,1]$, both stored as unconstrained logits and squashed at use. The continuous fields $a(\mathbf{x})$ and $m(\mathbf{x})$ in Eq. (1) are obtained by blending these per-disk attributes over the disks near $\mathbf{x}$ (described next).

Partition-of-unity blend. A fragment at $\mathbf{x}$ on face f gathers from $\mathcal{N}(f)$, the fixed set of the $K{=}16$ disks nearest that face. These disks are not restricted to f: the top-K set spans f and its neighboring faces, so the gather crosses face boundaries rather than being truncated per face. Each disk is weighted by its own Gaussian density at $\mathbf{x}$, evaluated in the disk's tangent frame:

$$w_k(\mathbf{x}) = \exp\Big(-\tfrac{1}{2}\big[(u_{k,1}/s_{k,1})^2 + (u_{k,2}/s_{k,2})^2\big]\Big), \quad \bar{w}_k(\mathbf{x}) = w_k(\mathbf{x}) \Big/ \sum_{j \in \mathcal{N}(f)} w_j(\mathbf{x}), \tag{4}$$

where $u_{k,j} = (\mathbf{x} - \mathbf{x}_k) \cdot \mathbf{t}_{k,j}$ is the fragment's in-plane offset from disk k's center along its tangent axis $\mathbf{t}_{k,j}$. The per-fragment fields are the resulting convex combination,

$$a(\mathbf{x}) = \sum_{k \in \mathcal{N}(f)} \bar{w}_k(\mathbf{x})\, a_k, \quad m(\mathbf{x}) = \mathrm{sigmoid}\Big(\sum_{k \in \mathcal{N}(f)} \bar{w}_k(\mathbf{x})\ \mathrm{logit}(m_k)\Big), \tag{5}$$

with a_k, m_k the squashed per-disk albedo and reflectivity. The weights form a partition of unity ($\sum_k \bar{w}_k = 1$), so each field is a *normalized* convex combination of the nearby disks rather than an alpha composite. This is the order-independence the representation was designed around (Sec. 1): the blend never asks which disk lies in front, so the coplanar depth ties cannot arise and the $\mathcal{O}(K)$ gather needs no sort.

3.3 Adaptive Environment Map

We describe the reflection lookup first without its learned parts, where it is ordinary environment mapping: a directional fetch and a weighted blend.

Fetching. Treat the environment as a classical equirectangular map. At a surface point with normal $\mathbf{n}$ seen from view direction $\mathbf{v}$, the mirror-reflected direction is $\boldsymbol{\omega}_r = 2(\mathbf{n} \cdot \mathbf{v})\,\mathbf{n} - \mathbf{v}$. We then convert to a longitude–elevation coordinate and bilinearly sample according to the direction.

Which map: near-convex parts. A single global map assumes all light arrives from infinitely far away, so it cannot reproduce the light an object casts on itself. We resolve this geometrically. Within a convex part there is no self-occlusion and no self-reflection, so all incident light is external and a single directional map is a good approximation; what a global map cannot capture is the light exchanged *between* parts. We therefore split the object into near-convex parts with a convex decomposition (Wei et al., 2022) and give each part its own map, so that light reaching a part from the rest of the object is absorbed into that part's map and read back as an ordinary far-field lookup. The approximation is that this inter-part light is treated as distant and position-independent within a part, and Sec. 4.3 measures what it recovers.

Growing the maps by error. We do not fix the number of maps in advance. Training begins with one map shared by the whole object. Periodically we attribute the current reconstruction error to parts and split off the highest-error *connected* region into a new child map, warm-started by copying its parent. Repeated up to a small budget (five maps in our experiments), this spends capacity where reflection error is largest, isolating the concave pockets a single map cannot explain.

Blending across borders. Hard per-face routing would leave visible seams at part boundaries. Instead, each face holds a soft membership over the maps and a fragment reads *every* map at $\boldsymbol{\omega}_r$:

$$E(\boldsymbol{\omega}_r, f) = \sum_p \beta_p(f)\, E_p(\boldsymbol{\omega}_r), \qquad \beta_p(f) = \mathrm{softmax}_p\big(-d_p(f)/\tau\big), \tag{6}$$

Table 1: Appearance quantitative on NeRF-Synthetic and Shiny-Blender datasets. Ranking (**1st** , 2nd , 3rd) is computed within the Fixed-Mesh block only, since these methods share our input mesh; the No-Mesh and Flexible-Mesh blocks reconstruct their own geometry and are shown as context, not ranked (numbers reported from RefGS (Yao et al., 2025)). The Fixed-Mesh rows (2DGS*, DNR, and Ours) are measured by us.

Constraint	Method	NeRF-Synthetic			Shiny-Blender		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
No Mesh (context)	Ref-NeRF (Verbin et al., 2022)	31.29	0.947	0.058	32.32	0.956	0.110
	3DGS (Kerbl et al., 2023)	33.30	0.969	0.030	30.37	0.947	0.083
	3iGS (Tang & Cham, 2024)	33.60	0.970	0.029	30.64	0.948	0.077
	ENVIDR (Liang et al., 2023)	28.13	0.953	0.068	32.88	0.969	0.072
	3DGS-DR (Ye et al., 2024)	31.02	0.962	0.047	33.94	0.971	0.059
Flexible Mesh (context)	GaussianShader (Jiang et al., 2024)	31.48	0.960	0.042	30.42	0.951	0.082
	RefGS (Yao et al., 2025)	33.20	0.966	0.036	34.80	0.973	0.056
Fixed Mesh	DNR (Thies et al., 2019)	32.01	0.959	0.043	27.50	0.931	0.097
	2DGS* (Huang et al., 2024a)	32.32	0.960	0.029	27.70	0.930	0.078
Fixed Mesh	Ours	**33.74**	**0.972**	**0.025**	**38.04**	**0.983**	**0.024**

where $d_p(f)$ is the distance from face f to the nearest face assigned to map p, and τ sets the blend width: β is $\approx$ one-hot inside a region and blends smoothly within $\sim \tau$ of a border, so the weighted fetch removes the seam. Each E_p stores a learned multi-channel feature rather than RGB, decoded into reflected color by the network of Sec. 3.4.

3.4 The FastKAN Reflection Decoder

The fields of Secs. 3.2 and 3.3 are tied together by a learnable decoder: it consumes a surface point, its viewing geometry, and the part-blended environment feature, and emits the view-dependent reflection $r(\mathbf{x}, \mathbf{v})$ of Eq. (1). The decoder is interchangeable. We use a Fast Kolmogorov–Arnold Network, which places a learnable activation on every *edge* rather than a fixed nonlinearity on every node (Liu et al., 2025), realized with Gaussian radial basis functions on a small fixed grid (Li, 2024); on our task it outperforms a parameter-matched MLP by 2.44 dB (Sec. 4.3). Where inference cost matters more than fidelity, an MLP or a precomputed lookup table can occupy the same slot.

Staged inputs. The decoder is a deferred, staged shader (Verbin et al., 2022): view-independent quantities are consumed before view-dependent ones, with two KAN layers per stage. (1) *Position*: the Fourier positional encoding of the hit point $\mathbf{x}$ (bounding-box normalized, six frequencies), which localizes appearance on the surface. (2) *Normal*: the real spherical harmonics $\mathrm{SH}(\mathbf{n})$ of the surface normal (degree 4) are appended. (3) *View*: the integrated directional encoding $\mathrm{IDE}(\boldsymbol{\omega}_r)$ of the reflected direction is appended together with $\mathbf{n}\cdot\mathbf{v}$; the IDE is $\mathrm{SH}(\boldsymbol{\omega}_r)$ attenuated by a learned per-point roughness, so a rougher surface queries the environment over a wider cone — a prefiltered, glossy lookup rather than a perfect mirror. This stage emits the material feature $\mathbf{h}$.

Environment fusion and output. A two-layer linear MLP fuses $\mathbf{h}$ with the part-blended environment feature $E(\boldsymbol{\omega}_r, f)$ of Eq. (6), and a linear head then reads the fused feature to produce a raw specular output, which is scaled by a learned specular tint in $[0, 1]^3$ (predicted from an earlier stage of the network separately) to give the view-dependent reflection $r(\mathbf{x}, \mathbf{v})$ of Eq. (1). Only the staged trunk is KAN. The color head is zero-initialized, so reflection starts at zero and is turned *up* only where the training views demand it.

4 Experiments

Novel View Synthesis We conduct experiments on NeRF-Synthetic (Mildenhall et al., 2020) and Shiny-Blender (Verbin et al., 2022) in a *fixed mesh setting*. We choose two fixed-mesh baselines: DNR (Thies et al., 2019) and a mesh-constrained 2DGS, denoted 2DGS* (Huang et al., 2024a). We initialize 2DGS* with four million 2D Gaussians uniformly on the surface and all gradients are projected onto the manifold defined by the mesh faces. We also report no-mesh methods (Ref-NeRF (Verbin et al., 2022), 3DGS (Kerbl et al., 2023), 3iGS (Tang & Cham, 2024), ENVIDR (Liang et al., 2023), 3DGS-DR (Ye et al., 2024)) and flexible-mesh methods (GaussianShader (Jiang et al.,

Table 2: **Ordering degeneracy.** Mean Shiny-Blender PSNR for 2DGS* with a fixed per-disk normal offset ε at initialization, versus the tied ε=0 baseline (Table 6). PSNR does not improve with ε.

Dataset	ε=0 (baseline)	10^{-5}	10^{-4}	10^{-3}
Shiny-Blender (PSNR↑)	27.70	27.63	27.59	27.66

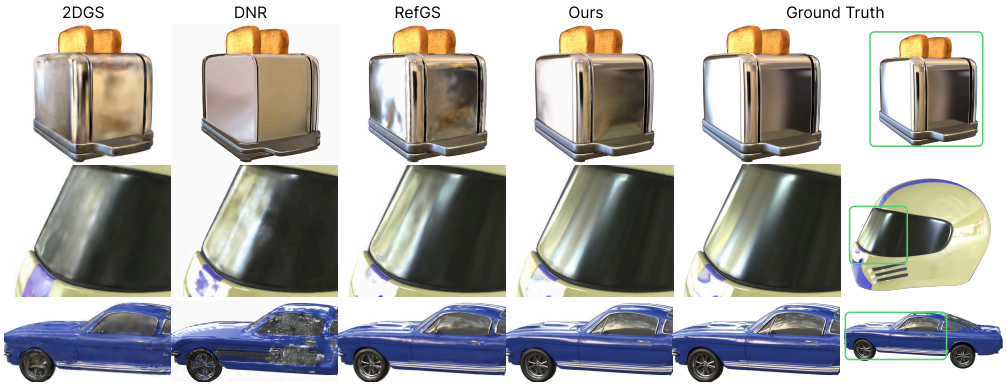

Figure 3: **Qualitative comparison on Shiny-Blender.** From left to right: the mesh-constrained baseline 2DGS*, DNR, the state-of-the-art RefGS, our method, and ground truth (GT); the rightmost column marks the magnified region on the full object. Our renders reproduce the sharp, high-frequency environment reflections on the metallic surfaces most faithfully; 2DGS* and DNR blur or smear them, RefGS recovers them less crisply.

2024), RefGS (Yao et al., 2025)) as reference. Note that these methods optimize geometry rather than taking it as given, so they are not directly comparable.

4.1 Ordering Degeneracy

Theorem B (App. A.1) predicts that a fixed normal offset cannot repair the ordering obstruction of Sec. 1: every finite offset still flips at some visible view, and the offset itself receives no learning signal. We confirm this on 2DGS*: displacing every disk by a fixed ε along the face normal at initialization, with no further normal motion during training, PSNR stays flat from ε=10^{-5} to 10^{-3} and matches the tied ε=0 baseline (Table 2).

4.2 Appearance Quality

Given the mesh, our appearance field reaches 38.04 dB PSNR on Shiny-Blender and 33.74 dB on NeRF-Synthetic (Table 1), the best result among fixed-mesh methods that share our input mesh; the no-mesh and flexible-mesh rows reconstruct their own geometry and are shown as context, not ranked against ours. The margin over the best context method (Table 7) tracks how reflective a scene is, from +6.12 dB on the near-mirror *ball* down to +0.45 dB on *teapot*, whose highlights are smallest. Qualitatively (Figure 3), our renders reproduce sharp, high-frequency reflections on metallic surfaces that 2DGS* and DNR blur.

4.3 Ablation

Table 3 follows the path by which the appearance model was built. The coplanar 2DGS* baseline suffers the depth-fighting of Sec. 1, and a purely diffuse mixture cannot represent view-dependent reflection, so both collapse on the reflective scenes. Per-primitive view-dependent color — spherical harmonics in 3DGS, or the anisotropic spherical Gaussians of Spec-Gaussian (Yang et al., 2024) — is strong for *floating* splats, where overlapping, depth-ordered primitives compose the appearance. On a single opaque surface that composition is unavailable: each point takes one order-independent blend, so a per-primitive spherical Gaussian with an MLP decoder only overfits the sparse views (34.24 dB), and dropping it barely changes the result (Table 3). The gains come instead from

Table 3: **Ablation on Shiny-Blender** along our development path (dataset means; **best** in bold). From the 2DGS* baseline we place the mixture on the manifold (2D GMM), add a per-primitive spherical-Gaussian (SG) reflection with an MLP decoder, drop SG, add our adaptive hierarchical environment maps (AdaMap), and finally replace the MLP with our KAN. The MLP and KAN decoders are matched in parameter count.

Variant	PSNR↑	SSIM↑	LPIPS↓
2DGS* (baseline)	27.70	0.930	0.078
2D GMM	31.40	0.952	0.057
+ SG + MLP	34.24	0.961	0.051
+ MLP + 1 EnvMap	34.37	0.963	0.049
+ MLP + AdaMap	35.60	0.971	0.038
+ KAN + AdaMap	**38.04**	**0.983**	**0.024**

Table 4: **Element-wise compaction.** Mean PSNR/SSIM/LPIPS on Shiny-Blender after compacting the disk field with NanoGS (Xiong et al., 2026), a training-free Gaussian-splat simplifier, at the fraction of disks kept. Because each disk is an ordinary, individually addressable splat, off-the-shelf splat-compaction tools apply directly; quality degrades gracefully even at $10\times$ compaction. Per-scene results in Table 10, App. A.4.

Kept	PSNR↑	SSIM↑	LPIPS↓
100%	38.04	0.9830	0.0238
50%	37.67	0.9799	0.0275
25%	37.11	0.9764	0.0324
10%	36.03	0.9700	0.0443

the adaptive hierarchical maps ($+1.2$ dB over a single global map) and from the KAN decoder ($+2.44$ dB over a parameter-matched MLP).

4.4 Efficiency

The full asset (10^6 disks, five environment maps, shader) occupies 48.3 MB on disk. At 800×800 we render at 91.7 FPS on NeRF-Synthetic and 68.6 FPS on Shiny-Blender on an NVIDIA H100 (CUDA 12.8), and at 76.8 FPS on Shiny-Blender on a consumer RTX 4090 (WSL2, Ubuntu 22.04), with a mean CUDA memory footprint of 1.74 GB (peak 2.75 GB), all under `torch.compile`.

4.5 Compaction

Because each disk is an ordinary splat rather than a bespoke primitive, we compact the baked asset with NanoGS (Xiong et al., 2026), a training-free Gaussian-splat simplifier, with no change to our representation or pipeline. Table 4 reports mean PSNR/SSIM/LPIPS on Shiny-Blender as we keep 100%, 50%, 25%, and 10% of the disks: at $10\times$ compaction, mean PSNR falls by only 2.0 dB ($38.04 \rightarrow 36.03$), with SSIM and LPIPS degrading similarly gracefully. Per-scene results (Table 10, App. A.4) show *teapot* and *ball* essentially unaffected even at $10\times$, while the more detailed *helmet* loses the most. Because compaction reuses tooling built for the Gaussian-splatting community rather than a bespoke method, our representation is compatible with that ecosystem by construction.

5 Conclusion and Discussion

We present automatic neural baking, a representation that composes directly with the traditional rendering pipeline: an element-wise "texture" whose domain is the mesh surface and whose blending is order-independent, so the asset needs no UV atlas and no depth sort and is evaluated per fragment at real-time rates. A 2D Gaussian mixture anchored in barycentric coordinates carries the view-independent fields, an adaptive hierarchical environment map the incident light, and a FastKAN decoder the view-dependent term. With this bridge in place, the next steps are engine integration, real captures, and element-level operations such as merging or editing disks directly.

Scope. Our method takes the mesh as input, is optimized per scene, and is evaluated on synthetic single-object benchmarks. Within a near-convex part, light from the rest of the object is treated as distant and position-independent, so strongly concave objects are the hardest case: *toaster* has our lowest absolute PSNR, even though we still lead every context method there (Table 7). Real-time evaluation is demonstrated in nvdiffrast rather than a shipped shader. We evaluate with the provided meshes; behavior on reconstructed meshes is not studied in our main experiments. Because the anchoring is linear in the vertices, appearance gradients also reach the mesh; App. A.3 uses this property to refine a reconstructed mesh as a demonstration, and a full study of refinement is left to future work.

6 AI DISCLOSURE

In this work, we used generative AI tools to assist with writing and editing the language of the paper, including grammar and LaTeX formatting. The ideas of the method were developed by the authors. We used generative AI tools (e.g., Claude Code) in an assistive capacity during code implementation, including evaluation scripts, visualization scripts, and debugging; generative AI tools also assisted in implementing the proposed method. For data cleaning and reformatting, generative AI tools assisted in writing preprocessing and dataloader code; they were not used to edit or generate the data itself. We also used generative AI tools to assist in developing theoretical models and conceptual frameworks, formulating mathematical claims, and writing proofs, including providing critical ingredients for the proofs.

We have not used generative AI tools to generate synthetic datasets, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, or to translate. Research direction, qualitative analysis, interpretation of results, and experiment design are the authors' own.

The authors have reviewed all AI-assisted work. All AI-assisted proofs and mathematical claims were checked by the authors, and AI-assisted code was tested for correctness.

We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

7 REPRODUCIBILITY

All datasets used, NeRF-Synthetic (Mildenhall et al., 2020) and Shiny-Blender (Verbin et al., 2022), are public benchmarks with standard splits. The training objective (Eq. (2), Adam optimizer, 30K iterations) is given in Sec. 3.1 and the mixture's hyperparameters in Sec. 3.2; the convex decomposition and adaptive environment-map budget in Sec. 3.3; and the FastKAN decoder architecture in Sec. 3.4. We will release our full codebase upon the paper's acceptance or publication.

REFERENCES

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

Brent Burley and Dylan Lacewell. Ptex: Per-face texture mapping for production rendering. *Computer Graphics Forum*, 27(4):1155–1164, 2008.

Yuhang Cao, Haojun Yan, and Danya Yao. OMeGa: Joint optimization of explicit meshes and Gaussian splats for robust scene-level surface reconstruction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

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*, pp. 16569–16578, 2023.

Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982.

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 (SIGGRAPH)*, pp. 303–312, 1996.

Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

Hao-Bin Duan, Miao Wang, Jin-Chuan Shi, Xu-Chuan Chen, and Yan-Pei Cao. Bakedavatar: Baking neural fields for real-time head avatar synthesis. *ACM Transactions on Graphics (ToG)*, 42 (6):1–17, 2023.

Wenhang Ge, Tao Hu, Haoyu Zhao, Shu Liu, and Ying-Cong Chen. Ref-neus: Ambiguity-reduced neural implicit surface learning for multi-view reconstruction with reflection. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4228–4237. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00392. URL https://doi.org/10.1109/ICCV51070.2023.00392.

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*, pp. 5354–5363, 2024.

Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5875–5884, 2021.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, 2024a.

Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. Nerf-texture: Synthesizing neural radiance field textures. *IEEE transactions on pattern analysis and machine intelligence*, 46(9):5986–6000, 2024b.

Rasmus Ramsbøl Jensen, Anders Lindbjerg Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 406–413. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.59. URL https://doi.org/10.1109/CVPR.2014.59.

Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3D gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Brian Karis. Real shading in Unreal Engine 4. In *ACM SIGGRAPH 2013 Courses: Physically Based Shading in Theory and Practice*, 2013.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. Rendering neural materials on curved surfaces. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–9, 2022.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*, pp. 362–371. ACM, 2002. doi: 10.1145/566570.566590.

Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM Transactions on Graphics*, 37(6): 247:1–247:13, 2018.

Ziyao Li. Kolmogorov-arnold networks are radial basis function networks. *arXiv preprint arXiv:2405.06721*, 2024.

Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. ENVIDR: Implicit differentiable renderer with neural environment lighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. NeRO: Neural geometry and BRDF reconstruction of reflective objects from multiview images. *ACM Transactions on Graphics (TOG)*, 42(4), 2023.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-Arnold networks. In *International Conference on Learning Representations (ICLR)*, pp. 70367–70413, 2025.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

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. In *European Conference on Computer Vision (ECCV)*, pp. 405–421, 2020.

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics*, 43(6), 2024.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8280–8290, 2022.

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics*, 22(3):376–381, 2003.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. Autocuts: Simultaneous distortion and cut optimization for UV mapping. *ACM Transactions on Graphics*, 36(6):215:1–215:11, 2017.

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. SplattingAvatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Pratul P. Srinivasan, Stephan J. Garbin, Dor Verbin, Jonathan T. Barron, and Ben Mildenhall. Nuvo: Neural UV mapping for unruly 3D representations. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhe Jun Tang and Tat-Jen Cham. 3iGS: Factorised tensorial illumination for 3D gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5491–5500, 2022.

Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. GaMeS: Mesh-based adapting and modification of Gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024. URL `https://arxiv.org/abs/2402.01459`.

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3D meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

Butian Xiong, Rong Liu, Tiantian Zhou, Meida Chen, Zhiwen Fan, and Andrew Feng. NanoGS: Training-free Gaussian splat simplification. In *European Conference on Computer Vision (ECCV)*, 2026.

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*, 2024.

Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, pp. 597–614, 2022.

Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-Gaussian: Anisotropic view-dependent appearance for 3D gaussian splatting. In *Advances in Neural Information Processing Systems*, 2024.

Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. Reflective gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2025.

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*, pp. 1–9, 2023.

Keyang Ye, Qiming Hou, and Kun Zhou. 3D gaussian splatting with deferred reflection. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 371–378, 2001.