LIGHT FIELD PRIMITIVE FOR NOVEL VIEW SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present **L**ight **F**ield **P**rimitives (**LFP**), a formulation for novel view synthesis that replaces the dense ray database (Levoy & Hanrahan, 1996) with a compact set of differentiable primitives in the classical two-plane parameterization. Each primitive condenses a group of rays into one learned record, and its response to a query is governed by how closely that query belongs to the group. Rendering a camera ray then reduces to compositing all responses it elicits, and a scene can be optimized directly from posed images and rendered in real time with rays. Beyond its competitive performance on standard benchmarks, the main advantage of LFP is structural: its primitives reside directly in the 4D ray space, so optical and appearance effects that are already operations on the light field become behaviors of a single shared renderer. With minimal changes to that renderer, LFP supports multi-scale anti-aliasing, defocus deblurring with refocusing, rendering for fisheye cameras, and even transparent object reconstruction with ray refraction, matching specialized frameworks that devote substantial machinery to these effects.

1 INTRODUCTION

In recent years, novel view synthesis (NVS) has been reshaped by neural radiance fields and point-based rendering. NeRF (Mildenhall et al., 2020) represents a scene as a continuous radiance field in 3D space and queries it along camera rays, and 3DGS (Kerbl et al., 2023) represents it as a set of optimizable 3D Gaussians and rasterizes them for real-time rendering. Despite their different designs, both paradigms anchor the scene in 3D space. A camera, however, does not measure a quantity at a point in space; it measures the radiance arriving along each of its rays. Their renderers must therefore mediate between the two, restating one side in terms the other can evaluate: NeRF turns a ray into point samples, the form a field takes values on; 3DGS restates each Gaussian as a screen-space footprint that a pixel can evaluate, and later variants restate the ray in each Gaussian's own frame, solving for where it crosses one (Huang et al., 2024; Moënne-Loccoz et al., 2024). Mechanisms differ, but each connects a ray measurement to a representation anchored elsewhere.

Under the ideal model where a pixel is represented by one ray, their optimizations can learn scene parameters that read out correctly through the mediations. But every departure from the ideal model is a statement about rays: a coarser resolution widens the rays a pixel gathers into aliasing; a finite aperture spreads a ray bundle into defocus. Neither statement can be translated directly into space-anchored scene variables in NeRF nor 3DGS, and each must instead be reintroduced by rebuilding the renderer with extra complexity: dedicated anti-aliasing filters for scaling (Yu et al., 2024a; Barron et al., 2021), explicit defocus models for aperture (Wang et al., 2025; Lee et al., 2024a).

These complexities can be dissolved naturally once the scene is indexed by rays. As promised by classical light field rendering (LFR) (Levoy & Hanrahan, 1996), which parameterizes radiance by the intersections of rays with two parallel planes, anti-aliasing and defocus deblurring reduce to their clean optical definitions: prefiltering the ray database naturally produces antialiased renderings (Levoy & Hanrahan, 1996); averaging rays admitted by a synthetic aperture gives post-hoc refocusing (Isaksen et al., 2000), and omitting that averaging automatically yields deblurring. Both departures can be settled in the coordinates the representation already carries. However, classical light fields have to be captured on a dense grid of rays, incurring prohibitive storage overhead and restricting novel views to a fixed viewing volume. Lacking an underlying continuous representation, they could only interpolate between recorded rays rather than being optimized against a set of posed images, remaining static lookup tables.

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.

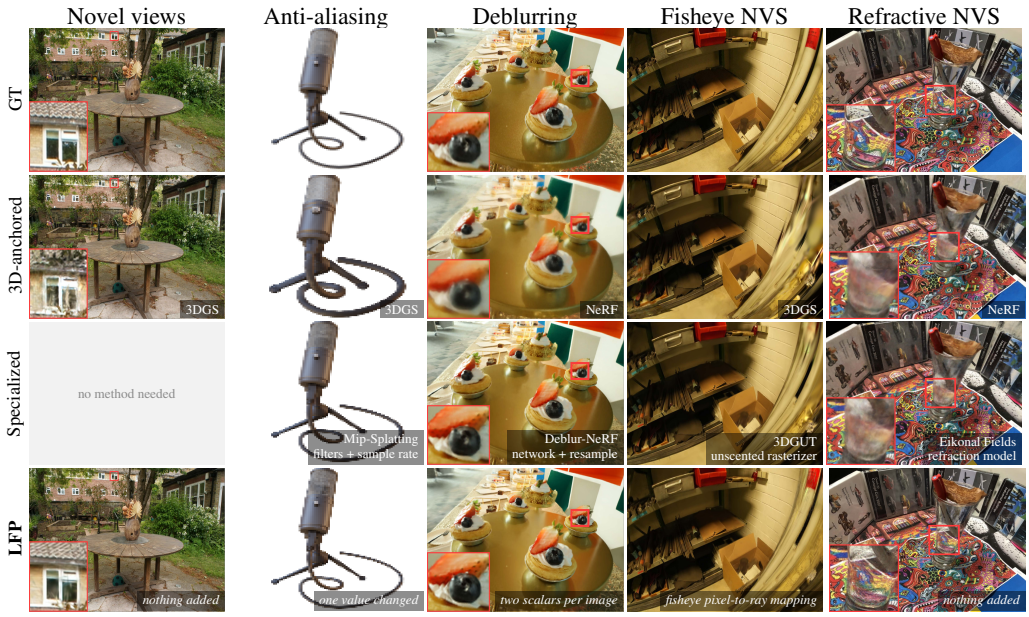


Figure 1: **One renderer, five capabilities.** Panels in the "Specialized" row are annotated with what each method adds over its base in the row above, and in "LFP" with what LFP adds over its base.

We bridge this historical divide with LFP, a formulation that brings the differentiable point-based optimization to the 4D ray space. LFP represents a scene as a compact set of primitives defined in the classical two-plane parameterization. Unlike classical light fields, which store radiance independently for each ray, a primitive is a learnable record shared by a group of rays: a 4D coordinate anchors one ray in the group, a disparity (Chai et al., 2000) supplies a coefficient, and together they define the affine coupling between the two plane coordinates of every ray in that group, the epipolar geometry (Bolles et al., 1987) written in these coordinates; an anisotropic covariance grades the response to a query (e.g., a camera ray) by how far it falls outside the ray group; and the record holds an opacity, a base radiance, and a directional term expressed with anisotropic spherical Gaussians (ASG) (Xu et al., 2013), so the radiance can differ from ray to ray within the group.

Meanwhile, since primitives are not parameterized by explicit 3D geometry, the projective splatting of 3DGS does not apply, requiring the interaction between a primitive and a query be established directly within the ray space. The interaction resides in a primitive's own parameters: its response to a query falls off as the query departs from the coupling that defines its ray group, measured under the covariance. Rendering then reduces to compositing all responses a query elicits, and it can be optimized from posed images and rendered in real time, retaining the optimizable nature of modern point-based methods while fully operating within the native coordinate system of light field.

As seen in Fig. 1, beyond performing competitively on standard NVS benchmarks (Barron et al., 2022; Knapitsch et al., 2017), the advantage of LFP is structural: statements about rays are settled by the ray representation itself without rebuilding the renderer. How many rays a pixel gathers is one such statement, and both departures, i.e., aliasing and defocus blur, ask only that a primitive be read over a wider tolerance. A primitive already carries that tolerance, so multi-scale anti-aliasing and defocus deblurring with post-hoc refocusing can be eased by simply supplying different values (see Sec. D), at a competitive quality (Yu et al., 2024a; Ma et al., 2022). Which ray a pixel stands for is another, costing even less: a fisheye capture changes how that ray is computed, but once computed it carries the same coordinates as any other, so the renderer handles it exactly as in the ideal model, while the point-based family must rebuild its projection for such a len (Wu et al., 2025). A third statement is what a ray carries. The grouping asks only that the radiance of a set of rays cohere, not what produced it, so a ray bent by glass is recorded like any other, and transparent objects can therefore be reconstructed without modeling the refraction explicitly (Bemana et al., 2022).

Main contributions of this work are threefold:

- We introduce LFP, an NVS method that brings the differentiable point-based optimization to the 4D ray space by modeling a scene as a set of primitives, each defined in the classical two-plane parameterization and holding one learned record for a group of rays.

- We exploit the affine coupling that defines a ray group both to gather the rays sharing a primitive's record and to grade a query against them, giving a response function that can render in real time within the 4D ray space.
- We demonstrate that LFP is effective on standard NVS benchmarks and, with minimal modifications to its renderer, can be extended to multi-scale anti-aliasing, defocus deblurring with post-hoc refocusing, fisheye rendering, and transparent-object reconstruction, at a quality comparable to frameworks specialized for each effect.

2 LIGHT FIELD REPRESENTATION

Rays as light field coordinates. As shown in Fig. 2 (a), we adopt the two-plane parameterization of classical light fields (Levoy & Hanrahan, 1996), where a unit normal $\mathbf{n} \in \mathbb{R}^3$ fixes a pair of parallel planes one unit apart: the uv plane at $\mathbf{n}^\top \mathbf{y} = 0$ and the st plane at $\mathbf{n}^\top \mathbf{y} = 1$. The planes share an orthonormal tangent basis $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^3$, with $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{n})$ defining the coordinate system. A ray can then be indexed by the tangential coordinates of its two intersections with the two planes. Concretely, for a camera with intrinsics $K \in \mathbb{R}^{3\times3}$ and rotation $R \in \mathbb{R}^{3\times3}$, the pixel at image coordinates (q_x, q_y) defines a ray that leaves the camera center $\mathbf{o}$ along the ray direction $\mathbf{d} = RK^{-1}(q_x, q_y, 1)^\top$ and is mapped to

$$\mathbf{x} = (u, v, s, t) = \big(\Pi(\mathbf{X}(\lambda_0)),\, \Pi(\mathbf{X}(\lambda_1))\big), \qquad \lambda_0 = -\frac{\mathbf{n}^\top \mathbf{o}}{\mathbf{n}^\top \mathbf{d}}, \qquad \lambda_1 = \lambda_0 + \frac{1}{\mathbf{n}^\top \mathbf{d}}, \tag{1}$$

with $\mathbf{x} \in \mathbb{R}^4$ the light field coordinate indexing the ray; $\mathbf{X}(\lambda) = \mathbf{o} + \lambda \mathbf{d}$ tracing the ray as λ advances from the camera, forward for $\lambda > 0$ and backward for $\lambda < 0$; λ_0 and λ_1 the ray parameters at which it meets the uv and st planes; and $\Pi(\mathbf{y}) = (\mathbf{e}_1^\top \mathbf{y},\, \mathbf{e}_2^\top \mathbf{y})$ extracting the tangential coordinates there. Every pixel can thus be mapped to a point in this ray space. Coordinates in that space are not all unrelated, and the structure that groups them, derived next, is what our primitives are defined on.

Epipolar geometry in ray space. Which rays should be represented by a shared record? They should be those whose radiance is coherent enough to be captured jointly by a single representation. Classical light field analysis (Chai et al., 2000) shows that such coherence is organized by epipolar geometry (Bolles et al., 1987): rays that see the same content are related across views by a constraint set by the disparity of that content. In two-plane coordinates this constraint takes a closed form. Consider a ray with coordinates (u, v, s, t) and a third parallel plane $\mathbf{n}^\top \mathbf{y} = z$, they intersect at

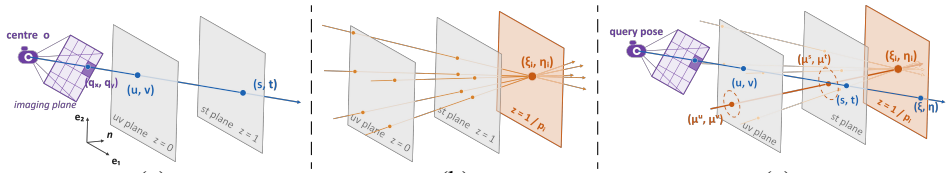
$$(\xi, \eta)^\top = (1 - z)(u, v)^\top + z(s, t)^\top, \tag{2}$$

which reduces to (u, v) at $z = 0$ and to (s, t) at $z = 1$. Eq. (2) holds for any z. Now assuming specific content at disparity p_i, it sits on the plane with $z = 1/p_i$, and every ray that sees it crosses that plane at the same point, whose tangential coordinate we write (ξ_i, η_i), as depicted in Fig. 2 (b). Rearranging Eq. (2) at that plane, they all satisfy $(s, t)^\top = (1 - p_i)(u, v)^\top + p_i(\xi_i, \eta_i)^\top$, which is an affine coupling between the two plane coordinates, determined by the disparity and the shared crossing through its coefficient $1 - p_i$ and offset $p_i(\xi_i, \eta_i)^\top$. This is the epipolar constraint made explicit: with v and t held fixed it is a line in the (u, s) plane, the line an epipolar-plane image (Bolles et al., 1987) shows for content at that disparity, with slope $1-p_i$. We call the rays satisfying one such coupling a ray group. Given the disparity, any one of its rays recovers the offset through Eq. (2), so a group is fully specified by its disparity and any one of its rays. Note that which disparity a group should take is not arbitrary: a light field is coherent only along particular disparity, and grouping rays at any other value mixes rays of unrelated radiance (Wanner & Goldluecke, 2012).

Primitives. LFP models a scene as a compact set of primitives, each attached to one ray group defined above and holding the record that group shares. Primitive i is specified by

$$\mathcal{P}_i = \{\ \boldsymbol{\mu}_i,\ p_i,\ \Sigma_i,\ o_i,\ \mathbf{c}_i\ \}, \tag{3}$$

where the first two entries identify the group, the third shapes its response to a query, and the last two form the shared record. Here $\boldsymbol{\mu}_i = (\mu_i^u, \mu_i^v, \mu_i^s, \mu_i^t) \in \mathbb{R}^4$ is the coordinate of one ray of the group, which we used as an anchor for the ray group, with (μ_i^u, μ_i^v) and (μ_i^s, μ_i^t) its intersections on the uv and st planes; $p_i > 0$ is the disparity of the group, setting the coefficient $1 - p_i$ of the affine coupling that holds among its rays, tying their st coordinates to their uv coordinates; $\Sigma_i \in \mathbb{R}^{2\times2}$ is an anisotropic covariance, setting how sharply the response falls off for queries outside the ray group; $o_i \in (0, 1)$ is the opacity of the record, modulating its peak contribution during compositing; $\mathbf{c}_i \in \mathbb{R}^3$ is its base radiance. Sec. 3.2 details a directional term that absorbs the residual variation within a group, while $\mathbf{c}_i$ carries the coherent part.



(a) (b) (c)

Figure 2: **Light field primitives for NVS. (a)** Two parallel planes index a ray by its crossings at them. **(b)** A ray group collects the rays that cross a third parallel plane at one point. **(c)** A primitive responds to a query (blue) by how far the query departs from its ray group, measured against the anchor ray it stores (thick orange), using their crossings at the initial two planes.

3 METHOD

3.1 RENDERING

Response. Given a query $\mathbf{x} = (u, v, s, t)$, e.g., a camera ray traced from a pixel, the response it elicits from primitive i is evaluated directly in ray space. Since the primitive and the query share the same light field coordinate system, no projection or screen-space linearization is required. Instead, we measure how far the query deviates from the ray group represented by the primitive.

At a fixed disparity p, the tangential coordinate at which a ray crosses the plane $z = 1/p$ partitions the ray space into distinct ray groups, and every ray belongs to one such group. The deviation of $\mathbf{x}$ from a ray group can thus be read off by comparing the differences between two groups, the one $\mathbf{x}$ falls in and the one the primitive represents, and we measure it under the partition at primitive i's own disparity p_i, through the residual

$$\mathbf{r}_i(\mathbf{x}) = p_i\big((\xi, \eta)^\top - (\xi_i, \eta_i)^\top\big) = (s - \mu_i^s, t - \mu_i^t)^\top - (1 - p_i)(u - \mu_i^u, v - \mu_i^v)^\top, \tag{4}$$

with (ξ, η) and (ξ_i, η_i) the tangential coordinates at which $\mathbf{x}$ and $\boldsymbol{\mu}_i$ meet the plane $z = 1/p_i$. The left form says what the residual is, which is the displacement between the two groups on that plane, scaled by p_i so that it measures deviation in angular rather than in world units; the right form says how it is evaluated, from the coordinates of the query and the parameters the primitive stores, with no actual intersection computed, as displaced in Fig. 2 (c). We convert the two-dimensional displacement into a scalar response using a Gaussian kernel:

$$G_i(\mathbf{x}) = \exp\Big(-\tfrac{1}{2}\mathbf{r}_i(\mathbf{x})^\top \Sigma_i^{-1} \mathbf{r}_i(\mathbf{x})\Big). \tag{5}$$

This formulation assigns the maximum response to rays belonging to the primitive's ray group and smoothly decreases with their deviation from the encoded coupling, where Σ_i controls the extent of the response. Note that nothing constrains Σ_i from below. If it is narrow along some direction, an appreciable response is confined to queries very close to the ray group in that direction, and once this range falls below the pixel spacing of a view, the primitive can be missed by that view and captured by another, which appears as flicker under a novel view. We therefore render with $\Sigma_i + \rho_i^2 \mathbf{I}$, where ρ_i is the width of half a pixel, measured as a displacement of the residual rather than of the image, so that no response is read more narrowly than a view can resolve. The same guard appears in EWA splatting, where the reconstruction kernel is convolved with a low-pass filter before it is sampled (Zwicker et al., 2002), and in 3DGS as a fixed dilation of the projected covariance (Kerbl et al., 2023). The conversion between the two widths is given in Sec. D. The response from each primitive is then accumulated through compositing.

Compositing. The opacity o_i caps primitive i's contribution. A query receives a discounted share,

$$\alpha_i(\mathbf{x}) = o_i\, G_i(\mathbf{x}) \le o_i, \tag{6}$$

with equality when the query lies in the primitive's ray group. Many primitives may respond to the same query, and their ordering affects the result, as in standard front-to-back alpha compositing (Max, 1995). Since the interaction with primitive i takes place on the plane $z = 1/p_i$ (Eq. (4) is a displacement on that very plane), the query meets primitive i at $\lambda_i = \lambda_0 + (\lambda_1 - \lambda_0)/p_i$ by Eq. (1), in which only p_i varies between primitives, so the disparities alone fix the sequence for each query, and a single sort by p_i serves the entire view. However, when a scene requires multiple light-field

coordinate systems (Sec. 3.2), disparities along different normals are no longer comparable, and the ordering must be resolved along a common axis. The camera axis is the natural choice, but a plane $z = 1/p_i$ does not have a single depth along it; a specific site on the plane must be chosen. We take the crossing point where the response peaks, i.e., (ξ_i, η_i) where the ray group is defined, which sits at $\xi_i \mathbf{e}_1 + \eta_i \mathbf{e}_2 + \mathbf{n}/p_i$ in the coordinate system $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{n})$, and read its depth in the camera coordinate system. This depth is the sorting key, applied once per view, with interactions behind the camera discarded. The query then accumulates

$$C = \sum\nolimits_{i=1}^{M} T_i \, \alpha_i \, \mathbf{c}_i + T_{M+1} \, \mathbf{c}_{\mathrm{bg}}, \qquad T_i = \prod\nolimits_{j<i} (1 - \alpha_j), \tag{7}$$

where T_i is the fraction the first $i-1$ primitives leave to the rest, M the number of ordered primitives, and $\mathbf{c}_{\mathrm{bg}}$ the background radiance receiving the remaining transmittance.

Note every step from Eq. (4) to Eq. (7) takes place in the ray space where the query already lives, so LFP needs no mediation: neither the ray marching of NeRF nor the projection of 3DGS. What a camera does to its rays, such as widening the bundle a pixel gathers, spreading it across an aperture, or bending the map from pixels to directions, therefore lands directly on quantities the renderer already reads, which is what Sec. 4 relies on for anti-aliasing, defocus deblurring, and fisheye NVS.

3.2 Model Details and Optimization

View-dependent appearance. The base radiance $\mathbf{c}_i$ is constant across a ray group. To capture the variation, we use 3 ASG lobes (Xu et al., 2013) for each group, which is natural for modeling view-dependent appearance (Yang et al., 2024). Each lobe responds to a small set of directions, so a narrow highlight costs a single lobe, whereas the spherical harmonics of 3DGS spread over all directions and need many high-order terms for the same effect. The two bandwidths of a lobe are independent, letting it stretch along one tangent direction and stay narrow along the other. The lobes are evaluated in a reflected direction, which requires an axis to reflect about. We use the anchor $\boldsymbol{\mu}_i$ as that direction, denoting it $\mathbf{m}_i$. Note nothing ties $\mathbf{m}_i$ to a surface, and the photometric loss is free to place it wherever it best explains how radiance varies across the group, so it acts as a shading normal rather than a geometric one. A unit query direction $\hat{\mathbf{d}}$ is mirrored about $\mathbf{m}_i$ into $\mathbf{h}_i = 2(\mathbf{m}_i \cdot \hat{\mathbf{d}})\mathbf{m}_i - \hat{\mathbf{d}}$. Following the standard ASG formulation, the actual radiance term becomes

$$\tilde{\mathbf{c}}_i(\hat{\mathbf{d}}) = \sigma\Big(\mathbf{c}_i + \sum\nolimits_k \mathbf{g}_i^k \, \max\big(\mathbf{h}_i^\top \boldsymbol{\omega}_i^k, \, 0\big) \, \exp\big(-\|(\mathbf{B}_i^k)^\top \mathbf{h}_i\|^2\big)\Big), \tag{8}$$

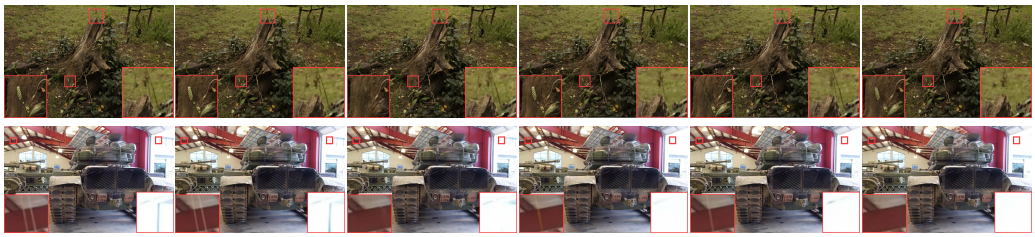
where lobe k carries a unit axis $\boldsymbol{\omega}_i^k$, an RGB amplitude $\mathbf{g}_i^k$, and a matrix $\mathbf{B}_i^k$ whose two columns are its tangent axes scaled by the square roots of the bandwidths. All of $\boldsymbol{\omega}_i^k$, $\mathbf{g}_i^k$, $\mathbf{B}_i^k$, and $\mathbf{c}_i$ are optimized per primitive. Each lobe brightens $\mathbf{c}_i$ over a narrow band of reflected directions.

Multi-atlas parameterization. A single plane pair cannot index rays that run nearly parallel to its planes, as their light field coordinates diverge. We therefore fix a set of atlases before optimization (16 atlas in practice), each with its own normal, tangent basis, and plane pair, with normals at the cluster centers of the training camera directions. A primitive keeps the atlas it is assigned, so the camera terms of Eq. (1) are computed once per camera and atlas rather than once per primitive.

Initialization. A ray group is specified by a plane parallel to the two parameterization planes and a tangential coordinate on that plane (Sec. 2). A point in space naturally provides such a pair: it determines the plane $\mathbf{n}^\top \mathbf{y} = z$ through its position along $\mathbf{n}$ and the tangential coordinate $\Pi(\mathbf{y})$ on that plane. A point therefore provides an initial identifier for a ray group, allowing a point cloud to initialize a set of primitives. We use the sparse output of structure from motion (Snavely et al., 2006) for our intialization. Note the initial points are not part of the representation and serve only as starting values, different from that in 3DGS. Each point is assigned to an atlas according to the alignment between a normal estimated from its neighbours and the atlas normal. In that atlas, the point gives $1/p_i = \mathbf{n}^\top \mathbf{y}$ and $(\mu_i^u, \mu_i^v) = (\mu_i^s, \mu_i^t) = \Pi(\mathbf{y})$. The resulting anchor ray runs along the atlas axis and therefore has identical tangential coordinates on the two planes. This anchor direction is shared by all primitives within the atlas and is not a per-point surface normal. The covariance is initialized isotropic, at a scale set from the local point spacing, so that a primitive's response falls off no faster than the gap to its neighbours. Since the group offset can be adjusted by optimizing (μ_i^s, μ_i^t), we fix (μ_i^u, μ_i^v) after initialization and optimize only (μ_i^s, μ_i^t), p_i, Σ_i, and the record.

Table 1: Average NVS results. Numbers are cited from (Kulhanek & Sattler, 2026) except those boxed with dash lines, which we ran ourselves. FPS is measured on our device for every method.

	Method	Space	Mediation	Mip-NeRF 360 (9 scenes) PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem	Tanks and Temples (21 scenes) PSNR ↑	SSIM ↑	LPIPS ↓	FPS	Mem
NeRF	Instant NGP	3D field	ray-marching	25.51	0.684	0.398	1.33	38.1MB	21.62	0.712	0.340	1.62	39.3MB
	NeRFStudio			26.39	0.731	0.343	1.72	68.4MB	22.04	0.743	0.270	3.88	68.4MB
	M-NeRF 360			27.68	0.792	0.272	0.12	34.4MB	23.43	0.784	0.215	0.27	34.4MB
GS	M-Splatting	3D point	projection	27.49	0.815	0.258	149	782MB	23.93	0.833	0.166	203	452MB
	3DGS			27.45	0.814	0.257	166	789MB	23.84	0.831	0.165	229	418MB
	3DCS			27.22	0.800	0.244	39.1	670MB	24.26	0.828	0.137	36.6	515MB
	LFP	4D ray	unnecessary	28.02	0.820	0.247	101	313MB	24.38	0.835	0.151	173	150MB



(a) GT (b) LFP (c) 3DCS (d) 3DGS (e) Mip-Splatting (f) Mip-NeRF 360

Figure 3: NVS examples on Barron et al. (2022) (*top*) and Knapitsch et al. (2017) (*bottom*).

Optimization. Rendering follows the tile-based design in (Kerbl et al., 2023): each primitive is assigned to the tiles it can affect, and each query composites the responses in its tile through Eq. (7). While 3DGS reads this assignment off the projected extent of a Gaussian, our primitives carry no explicit geometry to project, so we obtain it from the response itself: evaluating Eq. (1) at neighbouring pixels gives the rate at which the residual grows per pixel step, which converts Σ_i into a screen radius (see Sec. C for details). Every primitive is optimized against

$$\mathcal{L} = (1 - \lambda_{\mathrm{d}})\,\mathcal{L}_1 + \lambda_{\mathrm{d}}\,\mathcal{L}_{\text{D-SSIM}} + \lambda_{\mathrm{g}}\|\mathbf{g}\|^2, \tag{9}$$

where λ_d and λ_g are the weights, and the last term budgets the highlight amplitudes $\mathbf{g}$ of the appearance model so that the directional lobes do not take over color that belongs to the base. We also use densification during our optimization: cloning, splitting, and pruning for primitives. The criterion for selecting which primitives to densify cannot follow 3DGS, which uses the gradient w.r.t a projected 2D mean, since no such quantity enters our response. We accumulate instead the gradients from the parameters that define a ray group, i.e., $\boldsymbol{\mu}_i$ and p_i, as the densification signal.

4 Experiments

Datasets and Implementation. We evaluate LFP on 5 distinct setups: standard NVS on the 9 Mip-NeRF 360 scenes (Barron et al., 2022) and 21 Tanks and Temples scenes (Knapitsch et al., 2017), following protocol in Kulhanek & Sattler (2026); multi-scale anti-aliasing under the single-scale training and multi-scale testing protocol of Yu et al. (2024a), with zoom-in on the 9 Mip-NeRF 360 scenes and zoom-out on the 8 synthetic scenes of Blender (Mildenhall et al., 2020); defocus deblurring on the 10 real defocus scenes of Deblur-NeRF (Ma et al., 2022); fisheye NVS on the selected 6 scenes from ScaneNet++ (Yeshwanth et al., 2023) following Liao et al. (2024), and transparent-object reconstruction on the 4 glass and liquid scenes from (Bemana et al., 2022).

LFP is built in PyTorch with the renderer as a single CUDA kernel. The same kernel serves all setups, including the two covariance edits in anti-aliasing and deblur that are applied in the Python layer before the kernel is called, so anti-aliasing and defocus reuse the unchanged base renderer (see Sec. D for details). The loss weights are the same in every experiment, $\lambda_{\mathrm{d}} = 0.05$ and $\lambda_{\mathrm{g}} = 0.005$, and each scene trains for 30k iterations, evaluated at the last one, except the defocus setup, which trains for 20k. Within a setup no setting is tuned per scene, and every setup is initialized from COLMAP points unless otherwise mentioned. All experiments run on a single H100 GPU.

Standard NVS results. Tab. 1 shows results against representatives of both families (Müller et al., 2022; Barron et al., 2022; Tancik et al., 2023; Yu et al., 2024a; Kerbl et al., 2023; Held et al., 2025). Figures are taken from Kulhanek & Sattler (2026), which evaluates every method under

Table 2: Multi-scale rendering. A scene is trained once, at $n{=}1$, and rendered at four scales: zoom-in renders at $n\times$ the training resolution on Mip-NeRF 360, zoom-out at $1/n\times$ on Blender. All numbers are cited directly from Yu et al. (2024a), whose protocol we follow for LFP.

		PSNR ↑					SSIM ↑					LPIPS ↓				
	Method	$n{=}1$	$n{=}2$	$n{=}4$	$n{=}8$	Avg.	$n{=}1$	$n{=}2$	$n{=}4$	$n{=}8$	Avg.	$n{=}1$	$n{=}2$	$n{=}4$	$n{=}8$	Avg.
Zoom-in	Mip-NeRF	29.26	25.18	24.16	24.10	25.67	0.860	0.727	0.670	0.706	0.741	0.122	0.260	0.370	0.428	0.295
	3DGS	29.19	23.50	20.71	19.59	23.25	0.880	0.740	0.619	0.619	0.715	0.107	0.243	0.394	0.476	0.305
	3DGS + EWA	29.30	25.90	23.70	22.81	25.43	0.880	0.775	0.667	0.643	0.741	0.114	0.236	0.369	0.449	0.292
	Mip-Splatting	29.39	27.39	26.47	26.22	27.37	0.884	0.808	0.754	0.765	0.803	0.108	0.205	0.305	0.392	0.252
	LFP	29.75	27.44	26.48	26.19	27.47	0.879	0.794	0.737	0.753	0.791	0.100	0.213	0.321	0.409	0.261
Zoom-out	Mip-NeRF	33.08	33.31	30.91	27.97	31.31	0.961	0.970	0.969	0.961	0.965	0.045	0.031	0.036	0.052	0.041
	3DGS	33.33	26.95	21.38	17.69	24.84	0.969	0.949	0.875	0.766	0.890	0.030	0.032	0.066	0.121	0.063
	3DGS + EWA	33.51	31.66	27.82	24.63	29.40	0.969	0.971	0.959	0.940	0.960	0.031	0.024	0.033	0.047	0.034
	Mip-Splatting	33.36	34.00	31.85	28.67	31.97	0.969	0.977	0.978	0.973	0.974	0.031	0.019	0.019	0.026	0.024
	LFP	32.76	33.55	32.07	29.29	31.92	0.966	0.974	0.976	0.972	0.972	0.035	0.022	0.023	0.030	0.028

one protocol, except for the boxed rows, which we ran ourselves: the two point-based baselines, and Mip-NeRF 360 on Tanks and Temples, which the benchmark does not report. Our reevaluated 3DGS run lands within 0.02 dB of the cited one, indicating the two protocols agree. LFP is first on PSNR and SSIM on both datasets and second on LPIPS, performing competitively against many state-of-the-art methods. Quality aside, the two families differ in what they cost to render. The NeRF family stays below two frames per second, so the comparison to make is with the point-based methods, which rasterize as we do and differ in the space their primitives are defined in. Against 3DGS, LFP renders at about two thirds of the frame rate, still in real time, and holds under half the memory, and its variants make the same trade. Fig. 3 shows one scene from each dataset, where LFP preserves fine structures and image details while remaining visually comparable to current baselines.

What the ray-space parameterization buys is not in these numbers. Both NeRF and 3DGS families reach a primitive through a step the renderer must take before anything can be compared, marching the ray through the field or projecting each primitive to the image plane, and LFP needs neither: a query and a primitive are already written in the same coordinates, and their interaction is a subtraction. The capabilities this admits are the subject of the remaining experiments.

Multi-scale anti-aliasing. In this setting, a scene is trained at one resolution and rendered at others. Two directions are tested separately. A scene is trained once and rendered at $n \in \{1, 2, 4, 8\}$ times the training resolution for zoom-in, and at $1/n$ of it for zoom-out. In both directions the same trained scene serves every output resolution, and the covariance and opacity edits in Sec. D.1 are applied at render time with no gradient step. Results are in Tab. 2. At the training scale all methods are close, and the differences appear once the rendering resolution departs from it. 3DGS degrades sharply in both directions, since its screen-space dilation is tied to neither the sampling rate nor the primitive. The EWA filter recovers that loss but oversmooths, and Mip-NeRF stays stable across scales at a lower level throughout. LFP holds up in both directions and is close to the best Mip-Splatting.

However, what it takes to achieve the performance on either side is not comparable. LFP changes only the value of a term it already carries and rescales the opacity once, with no parameter added and nothing recomputed in training. Mip-Splatting needs two filters, one in world space and one in screen space, together with a per-primitive sampling rate recovered from all training cameras and maintained as primitives move and split, because a 3D Gaussian carries the same covariance whichever camera looks at it. The half-pixel width ρ_i of Sec. 3.1 already depends on the group's disparity and the camera being rendered, so each view supplies its own and neither filter is needed. Without either, LFP is competitive with Mip-Splatting across both directions, ahead on PSNR at the scales furthest from training and second on SSIM and LPIPS. In all, without any of these complexities, LFP is still competitive with Mip-Splatting, showing the potential of this new pipeline.

Defocus deblurring and refocusing. In this setting, training images are captured with a wide aperture and the held-out views are sharp, so a method must explain blurred observations with a scene that renders sharp. We apply the aperture term in Sec. D.2 when rendering the training views and drop it at test time, adding two scalars per training image, aperture and focal distance, which are optimized alongside the scene from a single initialization shared by all scenes. Results are in Tab. 3. Methods that do not model the aperture stay ineffective, neither 3DGS nor Mip-Splatting can ease the blur. Restoring their renderings afterwards with Restormer (Zamir et al., 2022) recovers part of

Table 3: Defocus NVS results on Ma et al. (2022).

	Method	PSNR ↑	SSIM ↑
General	3DGS	20.00	0.593
	3DGS+deblur	21.78	0.667
	NeRF	22.40	0.667
Specific	DOF-GS	23.48	0.742
	Deblur-NeRF	23.40	0.716
	Sharp-NeRF	22.82	0.694
	LFP	23.33	0.728

Table 4: Fisheye NVS results on Yeshwanth et al. (2023).

	Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	Instant NGP	26.53	0.847	0.273
	Nerfstudio	25.14	0.848	0.247
	M-NeRF 360	27.54	0.878	0.192
GS	3DGS	23.70	0.805	0.178
	FisheyeGS	28.14	0.894	0.152
	3DGUT	28.47	0.892	0.169
	LFP	28.52	0.889	0.167

Table 5: Transparent object NVS results on Bemana et al. (2022).

	Method	PSNR ↑	SSIM ↑	LPIPS ↓
General	NeRF	22.05	0.689	0.285
	Mip-Splatting	22.93	0.819	0.097
	3DGS	23.26	0.830	0.090
Specific	Eikonal Fields	21.54	0.700	0.258
	3DGRT	22.83	0.812	0.107
	RT-Splatting	23.37	0.827	0.091
	LFP	23.66	0.831	0.084

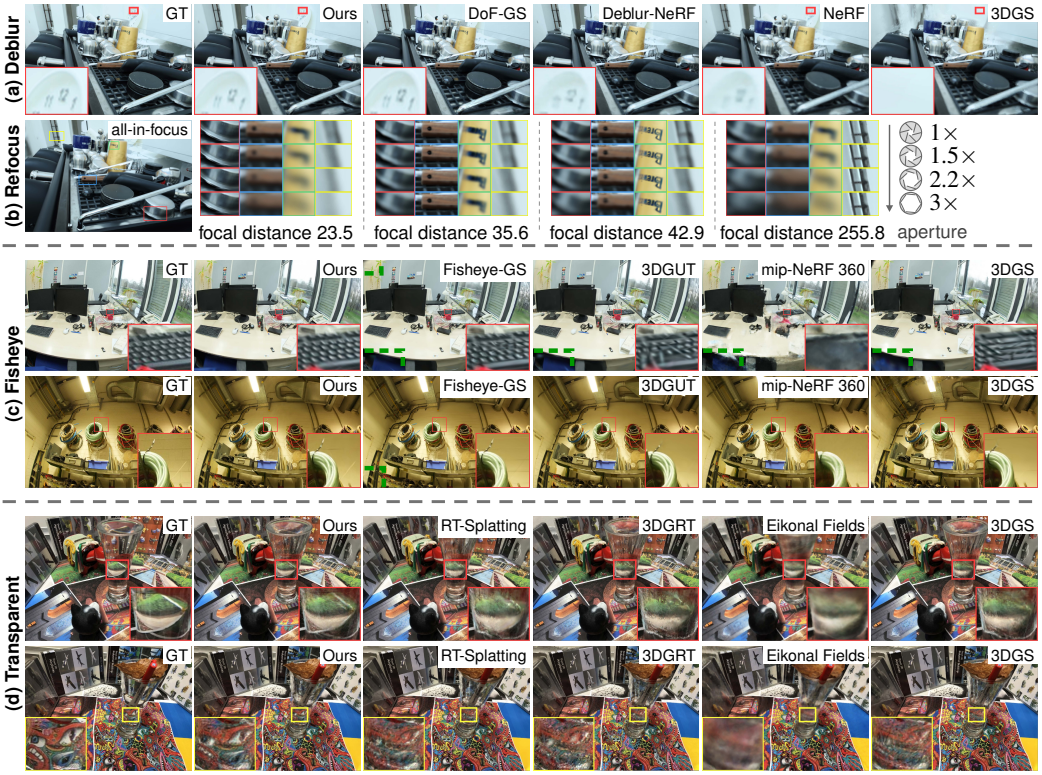


Figure 4: Qualitative comparisons across different settings.

the gap but works on images rather than on the representation. LFP is among methods built for this task, ranking third on PSNR and second on SSIM. Fig. 4 (a) shows the deblurred results: LFP recovers sharp structures and fine image details from the blurred observations, while the general-purpose baselines retain noticeable blur and the specialized methods produce comparable sharpness.

What this takes on either side is again not comparable. Those specific methods carry the aperture in the operator or attached to the primitives: the blur is predicted by an auxiliary network (Ma et al., 2022), guided by a prior computed over the training images (Lee et al., 2024a), or carried per primitive through the projection with the losses that keep it consistent (Wang et al., 2025). Differently, in LFP, the whole model of the aperture is simply two scalars per training image, with no auxiliary network and no added loss. What changes between the two settings is only when that term is applied: it is present when rendering the blurred training views and dropped at test time.

Refocusing then costs nothing further. Where the blur is fitted per primitive or per view, changing the focus after training means fitting it again. DOF-GS also supports post-capture refocusing (Wang et al., 2025), but its aperture modeling remains part of the rendering machinery. In LFP the aperture never enters the representation: the two scalars belong to the camera, and rendering at other values is the same closed-form update evaluated differently (Sec. D). Fig. 4 (b) shows a refocusing sweep at different focal distances. LFP moves the sharp region consistently with the selected focus, producing the expected transition from near to far objects while preserving the appearance of the scene.

Fisheye NVS. In this setting, the calibrated lens departs far from a linear projection. Nothing is added for the setting: the pixel-to-ray step of Eq. (1) follows the fisheye model instead of a pinhole

one, and every subsequent stage, the residual, the response, and the compositing, is the one used everywhere else. Results are in Tab. 4. The two families separate here for a reason: a radiance field is queried along whatever rays it is given, so the fisheye model enters only where the rays are generated and the NeRF family also handles the setting without modification (Müller et al., 2022; Tancik et al., 2023; Barron et al., 2022). A projection cannot be given rays. 3DGS has to be run on undistorted images instead, which discards the periphery the lens was chosen for and resamples what remains, and it falls well behind the rest of its family (Liao et al., 2024; Wu et al., 2025); the methods built for this setting keep the raw capture, either by replacing the projection with one the lens defines (Liao et al., 2024) or by propagating each primitive through the distortion (Wu et al., 2025). Both rebuild the step that reaches a primitive, and both are specific to the lens model they were written for. LFP leads on PSNR and follows closely on the other two, with no stage of the renderer changed: a ray is computed differently and then indexed by the same four coordinates as any other. In this respect LFP holds both sides of the table at once, taking rays as they come, as a field does, and rasterizing them in real time, as the point-based family does. Fig. 4 (c) compares two fisheye scenes against the baselines. While the central regions highlighted in the red boxes are reconstructed comparably well, some baselines exhibit noticeable artifacts or loss of detail toward the image boundaries, particularly in the corner regions highlighted in green. In contrast, LFP maintains consistent reconstruction quality across both central and peripheral regions.

Transparent objects. Through a refractive interface, rays reaching the camera are bent before measurement, so the observed radiance is no longer organized around the physical location of the object. Nevertheless, the refracted rays still exhibit coherent radiance across views. Sec. 2 defines primitives by grouping rays with coherent radiance and makes no assumption about the source of this coherence. Consequently, the ray groups formed in a refractive region are identical in definition to those elsewhere. A refracted ray is therefore represented in exactly the same way as any other ray, and the method applies without modification. Tab. 5 reports results on scenes with transparent objects. Methods that model the refraction do not clearly gain from it: the eikonal formulation traces rays through a recovered index-of-refraction field (Bemana et al., 2022) and stays below the methods that ignore the bending, while 3DGRT (Moënne-Loccoz et al., 2024) and RT-Splatting (Shi et al., 2026), which follow bent rays against the Gaussians, match 3DGS rather than exceed it. LFP is ahead of these arts, though by margins that leave refraction far from solved. The point is not the size of the lead but where it comes from: LFP adds nothing but still offers better results than those specific methods, suggesting that refraction may be less a modeling problem than a question of what a representation is indexed by. Fig. 4 (d) shows two scenes cropped to the refractive regions. Despite without explicit modeling, LFP preserves the contours and internal appearance of the glass and liquid objects, while the baselines show stronger artifacts and loss of detail in these regions.

5 Discussion and Conclusion

Limitations. Two limits follow from what the representation leaves out. LFP records rays rather than matter, so nothing in it is a surface, and tasks that need an explicit geometry are not served by the representation as it stands. The second is a debt of the parameterization. A single pair of planes cannot index rays that run nearly parallel to it, so a scene is given several atlases. Their normals are chosen from the training cameras before optimization and never change, and a primitive keeps the atlas it was assigned, so the coverage a scene has follows the capture rather than the content. Primitives in different atlases also fall back on a depth read in the camera frame for their ordering. A parameterization that indexes every direction without partitioning them would remove the second; the first is a consequence of recording a light field rather than a scene.

Conclusion. We present LFP, which represents a scene as a compact set of differentiable primitives in the classical two-plane parameterization. Each primitive holds one learned record for a group of rays and answers a query by how far the query departs from the coupling that defines the group, without projecting a primitive, intersecting one, or marching along a ray. A scene is optimized from posed images and rendered in real time, as with point-based methods. Departures from the ideal imaging model are settled in the coordinates the primitives are written in: a wider tolerance gives multi-scale anti-aliasing and defocus deblurring with post-hoc refocusing, a fisheye capture changes only how a ray is computed, and refraction asks for nothing, in each case matching methods built for that effect alone. Taken together, these are not four methods but one representation, and they suggest that ray space is a place to anchor a scene that the NeRF and 3DGS families have left open.

AI USE STATEMENT

We used generative AI tools to assist with writing: drafting and revising prose in the main paper and the supplementary material, improving clarity and grammar, and formatting tables and figure captions in LaTeX. Every claim, number, equation, and citation in the resulting text was written or verified by the authors against the implementation and the cited sources. We used generative AI tools to assist with code, specifically, e.g. evaluation scripts. Such code was read and tested by the authors. We did not use generative AI tools for research ideation or experimental design. Retrieval of related work was carried out by the authors; where a tool was used to surface candidate references, each was located and read in the original before being cited.

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

REFERENCES

Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding. In *CVPR*, 2022.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023.

Mojtaba Bemana, Karol Myszkowski, Jeppe Revall Frisvad, Hans-Peter Seidel, and Tobias Ritschel. Eikonal fields for refractive novel-view synthesis. In *SIGGRAPH*, 2022.

Robert C. Bolles, H. Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–55, 1987.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001.

Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *SIGGRAPH*, 2000.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *ECCV*, 2022.

François Darmon, Lorenzo Sinha, Ali Osman Ulusoy, and Thomas Brox. Robust gaussian splatting. arXiv preprint arXiv:2404.04211, 2024.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH*, 1996.

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. In *CVPR*, 2025.

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

Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *SIGGRAPH*, 2000.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):1–14, 2023.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 36(4):1–13, 2017.

Jonas Kulhanek and Torsten Sattler. NerfBaselines: Consistent and reproducible evaluation of novel view synthesis methods. In *NeurIPS*, 2026.

Byeonghyeon Lee, Howoong Lee, Usman Ali, and Eunbyung Park. Sharp-NeRF: Grid-based fast deblurring neural radiance fields using sharpness prior. In *WACV*, 2024a.

Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. In *ECCV*, 2024b.

Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996.

Zimu Liao, Siyan Chen, Rong Fu, Yi Wang, Zhongling Su, Hao Luo, Li Ma, Linning Xu, Bo Dai, Hengjie Li, et al. Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras. *arXiv preprint arXiv:2409.04751*, 2024.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 2024.

Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-NeRF: Neural radiance fields from blurry images. In *CVPR*, 2022.

Nelson Max. Optical models for direct volume rendering. *IEEE TVCG*, 1(2):99–108, 1995.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 38(4):1–14, 2019.

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 *ECCV*, 2020.

Nicolas Moënne-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 TOG*, 43(6):1–19, 2024.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022.

Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005-02, Stanford University, 2005.

Cheng Peng, Yutao Tang, Yifan Zhou, Nengyu Wang, Xijun Liu, Deming Li, and Rama Chellappa. BAGS: Blur agnostic gaussian splatting through multi-scale kernel modeling. In *ECCV*, 2024.

Ji Shi, Xianghua Ying, Bowei Xing, Ruohao Guo, and Wenzhen Yue. Rt-splatting: Joint reflection-transmission modeling with gaussian splatting. In *CVPR*, 2026.

Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021.

Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*. 2006.

Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *CVPR*, 2022.

Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021.

Yujie Wang, Praneeth Chakravarthula, and Baoquan Chen. DOF-GS: Adjustable depth-of-field 3d gaussian splatting for post-capture refocusing, defocus rendering and blur removal. In *CVPR*, 2025.

Sven Wanner and Bastian Goldluecke. Globally consistent depth labeling of 4D light fields. In *CVPR*, 2012.

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021.

Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. In *CVPR*, 2025.

Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. DoF-NeRF: Depth-of-field meets neural radiance fields. In *ACM MM*, 2022.

Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. Anisotropic spherical gaussians. *ACM TOG*, 32(6):1–11, 2013.

Ziyi Yang, Xinyu Gao, Yang-Tian Sun, Yi-Hua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. In *NeurIPS*, 2024.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *ICCV*, 2023.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3d gaussian splatting. In *CVPR*, 2024a.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM TOG*, 43(6):1–13, 2024b.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM TOG*, 37(4):1–12, 2018.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA volume splatting. In *IEEE Vis*, 2001.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA splatting. *IEEE TVCG*, 8(3):223–238, 2002.

A Related Work

Novel view synthesis (NVS). NeRF (Mildenhall et al., 2020) and 3DGS (Kerbl et al., 2023) families dominate current NVS task, and they operate on different sides of the 3D-to-2D mapping: the former by integrating a field along the ray, the latter by projecting 3D Gaussians onto the image plane.

Specifically, NeRF stores a scene as a continuous radiance field indexed by 3D position and view direction. A pixel is rendered by marching along its camera ray and integrating the field at the sample points. Later work keeps that operator and changes what sits at the samples. Hash grids, sparse voxels and tensor factorizations replace the coordinate network and cut training from days to

minutes (Müller et al., 2022; Fridovich-Keil et al., 2022; Chen et al., 2022). A second line makes the field scale-aware, encoding the cone a pixel subtends instead of a point on the ray, and carries that encoding to unbounded scenes and to feature grids (Barron et al., 2021; 2022; 2023); 3DGS regards a scene as a set of anisotropic Gaussians with opacity and view-dependent color. Each Gaussian is projected onto the image plane, where an affine approximation of the projection gives its screen-space footprint, following (Zwicker et al., 2001; 2002). Sorting and alpha-compositing the projected Gaussians inside a tile-based rasterizer gives real-time rendering, and adaptive density control grows and prunes the set during optimization. Later work revises the Gaussians or the stages around it, replacing the footprint with a ray-disk intersection (Huang et al., 2024), adding an opacity field for surfaces (Yu et al., 2024b), or decoding attributes from anchor points (Lu et al., 2024).

These two families differ in their rendering mechanisms, yet both anchor the scene in 3D space and rely on mapping operators to bridge that space with the sensor. We introduce LFP as a third option. LFP inherits the sparse differentiable optimization and tile-based rasterization of point-based methods, but defines its primitives natively in 4D ray space. Rather than projecting 3D primitives onto the image plane or integrating a field along a ray, LFP couples each primitive to a camera ray through an epipolar response function evaluated directly in ray space.

Light field rendering (LFR). LFR records radiance as a function of rays rather than matter that emits it (Levoy & Hanrahan, 1996; Gortler et al., 1996). A ray is parameterized by its intersections with two parallel planes, and a novel view is rendered by tracing each camera ray through those planes, querying the corresponding 4D coordinate, and resampling the stored radiance from its discrete neighbors. Because the representation is an explicit table of rays, optical effects reduce to operations on that table: integrating the rays admitted by a synthetic aperture yields depth of field and post-capture refocusing (Isaksen et al., 2000; Ng et al., 2005); prefiltering the table before resampling suppresses aliasing (Levoy & Hanrahan, 1996).

The cost of the simplicity is density. Alias-free reconstruction requires a sampling rate fixed by the scene depth range (Chai et al., 2000), so the table is large and views are bounded by the captured planes. Unrecorded rays can only be interpolated from neighbors, with hand-designed weights for unstructured capture (Buehler et al., 2001). The table is also static: it cannot be optimized against a sparse set of images. Learned successors escape the density at the cost of the explicit table. Layered representations collapse the 4D ray record into a stack of fronto-parallel depth planes (Zhou et al., 2018; Mildenhall et al., 2019; Wizadwongsa et al., 2021): the radiance of a ray is assembled by compositing the planes it pierces, and the continuous ray coordinate is discretized into plane indices and pixel grids. Neither an aperture integral nor a 4D prefilter can act on such a stack ray by ray, so both effects must be approximated with per-plane warps and blur kernels. Neural light fields compress the entire table into the weights of a network that maps ray coordinates to color (Sitzmann et al., 2021; Attal et al., 2022); with no explicit ray set left to integrate or prefilter, the effects either emerge as regularities learned from the training data or must be simulated by many network queries per pixel. Generalizable renderers sample source views along epipolar lines and blend their features (Suhail et al., 2022; Wang et al., 2021), and feed-forward models predict primitives directly (Charatan et al., 2024). In each case, the optical effects are no longer manipulations of an explicit ray structure, but behaviors recovered through network capacity or auxiliary modules.

LFP restores what these successors gave up. The dense table returns as a sparse set of differentiable primitives, each holding one learned record for a whole group of rays rather than one sample for one ray, so the light field stays explicit yet is no longer dense, and it can now be optimized against posed images. The effects therefore come back as manipulations of the representation: defocus is again an integration over rays, and anti-aliasing is again a prefiltering of them. Epipolar geometry changes role along the way, from a depth cue or a view-sampling rule to the relation that gathers the rays sharing a primitive's record. This keeps the optical advantages of the light field without its density burden, with no separate mechanism for any effect.

B MODEL DETAILS

Fields of a primitive. Sec. 2 specifies a primitive by $\{\boldsymbol{\mu}_i, p_i, \Sigma_i, o_i, \mathbf{c}_i\}$ and Sec. 3.2 adds the directional lobes. Three of these are constrained, and none is optimized in the form it is used. The disparity is stored as $\tilde{p}_i$ and decoded by $p_i = \exp \tilde{p}_i$, which keeps it positive without bracketing its range. The opacity is stored as a logit and passed through a sigmoid, giving $o_i \in (0, 1)$. The covariance is stored as three scalars with disjoint geometric roles, a log scale r_i, a log aspect ratio

Table 6: Fields of a primitive i. Floats are per primitive, with $K{=}3$ lobes.

Field	Stored as	Floats	Role
$\boldsymbol{\mu}_i$	$(\mu_i^u, \mu_i^v, \mu_i^s, \mu_i^t)$	4	anchor ray; (μ_i^u, μ_i^v) frozen after init
p_i	$\log p_i$	1	disparity of the group; coefficient of the coupling
Σ_i	(r_i, a_i, θ_i)	3	extent of the response, via Eq. (10)
o_i	logit	1	peak contribution in compositing
$\mathbf{c}_i$	logit RGB	3	base radiance of the record
$\mathbf{g}_i^k$	signed RGB	$3K$	amplitude of lobe k
$\boldsymbol{\omega}_i^k$	unit vector	$3K$	axis of lobe k
$\mathbf{B}_i^k$	$(\theta_i^k, \ell_i^k, a_i^k)$	$3K$	tangent frame and the two bandwidths
atlas index	integer	1	selects the plane pair; not optimized
Total, $K{=}3$		40	

a_i, and an angle θ_i, from which

$$\Sigma_i = \mathbf{R}(\theta_i)\,\mathrm{diag}\big(e^{2(r_i+a_i/2)},\, e^{2(r_i-a_i/2)}\big)\mathbf{R}(\theta_i)^\top, \qquad \mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{10}$$

symmetric positive definite for any value the three take, so no projection or clipping is needed during optimization. The two exponentials are the squared half-axes of the response, and $\mathbf{R}(\theta_i)$ orients them in the tangent plane. The renderer consumes the Cholesky factor of Σ_i, which Eq. (10) produces in closed form; the factor is clamped from below at 10^{-4} on its diagonal, the same bound the kernel enforces.

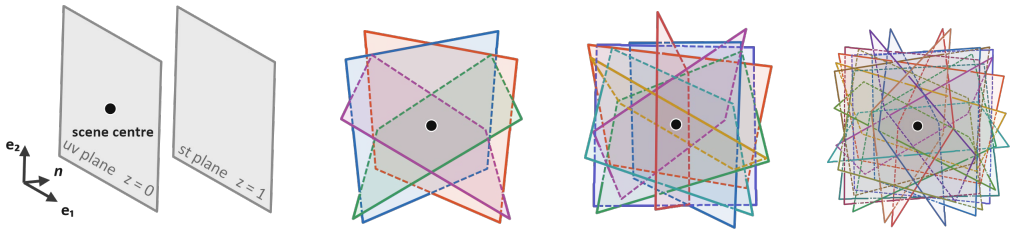
Each lobe of the appearance model is stored likewise. Its amplitude $\mathbf{g}_i^k$ is a signed RGB triple and its axis $\boldsymbol{\omega}_i^k$ a vector normalized before use. The matrix $\mathbf{B}_i^k$ of Eq. (8), whose columns are the two tangent axes scaled by the square roots of the bandwidths, is assembled from three scalars: an angle θ_i^k that rotates the tangent frame about $\boldsymbol{\omega}_i^k$, and two log bandwidths kept as a mean and a ratio, $\lambda_x^k = \exp(\ell_i^k + a_i^k/2)$ and $\lambda_y^k = \exp(\ell_i^k - a_i^k/2)$. Splitting scale from anisotropy in this way, for both Σ_i and the lobes, keeps the two directions of a covariance from competing through a shared parameter.

A primitive also carries an atlas index, assigned at initialization and fixed thereafter. It is not optimized and enters nothing but the choice of which $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{n})$ and which precomputed camera terms of Eq. (1) to read. Tab. 6 lists every field with its role and its cost.

Adaptive control of the primitive count. When primitives are initialized from a sparse structure from motion cloud we follow the schedule of 3DGS (Kerbl et al., 2023), cloning and splitting every 100 iterations between iteration 500 and $10\,000$, and resetting opacities to 0.06 every 3000 iterations between iteration 3000 and $10\,000$. The criterion is the one described in Sec. 3.2: gradients are accumulated on $\boldsymbol{\mu}_i$ and p_i, the parameters that place a ray group, since no projected 2D mean enters our response, and a primitive is selected when the running average exceeds 2×10^{-5}. A selected primitive is cloned when its larger radius stays below 0.005 of the scene extent and split otherwise.

A clone keeps the shape, orientation and atlas of its parent and is displaced inside the st plane: each of the two coordinates receives an independent Gaussian offset of standard deviation $\frac{1}{2}\max(r_i^x, r_i^y)$, half the primitive's own larger half-axis, so the pair separates by roughly its own width instead of starting as a duplicate. Parent and child each keep half the parent's opacity, which leaves the tangential coverage unchanged, and the next opacity reset removes whichever of the two the photometric loss does not support. Because the child sits elsewhere in the scene, its base radiance is not inherited but resampled: the child's centre is projected into the 20 most recent training views and read off the ground truth, falling back to the parent's value where it is not visible. The directional lobes are inherited unchanged.

A split replaces the parent by two children that inherit its record, its atlas and its base radiance. It is area-conserving and de-elongating: the children keep the shorter radius and halve the longer one, so an elongated primitive loses one factor of two in aspect and takes $1/\sqrt{2}$ of the parent's scale, while an already round one shrinks isotropically by that same factor.



(a) 1 atlas (b) 4 atlas (c) 8 atlas (d) 16 atlas

Figure 5: **Atlas construction.** A single atlas, shown in (a), is the classical two-plane parameterization of LFR (Levoy & Hanrahan, 1996): a uv plane through the scene centre and an st plane one unit along its normal $\mathbf{n}$. Only the uv planes are drawn in (b) - (d), each carrying its own st plane one unit away. Every atlas passes through the scene centre, so they differ only in orientation. The orientations are spread evenly here for legibility, and they are the K-means centres of the training camera directions in implementation.

Primitives are pruned when their opacity falls below 0.005, and additionally when their larger radius exceeds 0.1 of the scene extent.

Atlas construction. Atlas normals are chosen before optimization from the forward directions of the training cameras, and are fixed thereafter. The vertical axis is not assumed: it is taken as the direction of least variance of that distribution, so a scene is handled the same way whichever world axis happens to be up. The normals are then the centres of a k-means clustering of the camera directions, and we use 16 clusters on every benchmark. Fig. 5 shows our atlas construction.

A point can belong to an atlas only when it lies in front of that atlas's uv plane, as otherwise the disparity it would receive is not positive. Among the atlases that remain, each point of the initialization cloud takes the one whose normal best aligns with a surface normal estimated from its 10 nearest neighbours by local PCA, with alignment measured by the absolute dot product, since a surface normal and its opposite describe the same plane. Once made, the assignment is kept for the rest of training. Because the atlas count is small and each primitive's atlas is fixed, the whole atlas table is built once per frame and held in shared memory for the whole tile: the basis $(\mathbf{n}, \mathbf{e}_1, \mathbf{e}_2)$ of each atlas, which depends on neither the camera nor the query, together with the projections $\mathbf{n}^\top \mathbf{o}$, $\mathbf{e}_1^\top \mathbf{o}$, $\mathbf{e}_2^\top \mathbf{o}$ of the camera centre, which depend on the camera alone. Note that if each primitive carried its own plane pair, the same quantities would be per-primitive, and neither the computation nor the memory traffic could be amortized over a tile.

C TILE ASSIGNMENT WITHOUT PROJECTION

What has to be replaced. A tile-based rasterizer needs, for each primitive, the set of pixels whose queries can receive an appreciable response from it, so that the primitive is listed only in the tiles it can affect. In 3DGS this set is read off the projection: a 3D Gaussian is mapped to a 2D Gaussian on the image plane by an affine approximation of the projection (Zwicker et al., 2002), and the screen extent follows from the projected covariance. LFP has no such object to project. A primitive is a ray group and a record; its Σ_i weighs the residual of Eq. (4), a displacement between two-plane coordinates and not a screen quantity. What is available instead is the response as a function of the pixel, and both quantities the rasterizer needs are read from that function directly: its centre is the zero of the residual, and its extent is the residual's slope.

The residual as a function of the pixel. A pixel $\mathbf{x}(x, y)$ gives a query by Eq. (1), so composing the two makes the residual Eq. (4) a function of the pixel, which we write $\mathbf{r}_i(\mathbf{x})$. It vanishes exactly for the queries that belong to primitive i's ray group, and a view contains one such query: the one passing through (ξ_i, η_i) on the plane $z = 1/p_i$, the tangential coordinate the group is defined by. We write $\mathbf{x}^{\text{ref}}$ for its image coordinates and call it the reference position. Since (ξ_i, η_i) is determined by $\boldsymbol{\mu}_i$ and p_i, that query is known, leaving the camera centre towards $\xi_i \mathbf{e}_1 + \eta_i \mathbf{e}_2 + \mathbf{n}/p_i$, so inverting the pixel-to-ray map of Eq. (1) gives $\mathbf{x}^{\text{ref}}$ directly, with no search over pixels, as written out below. It is where the response of primitive i reaches one, its largest value.

Away from $\mathbf{x}^{\text{ref}}$ the residual is not affine in (x, y): the ray parameters λ_0, λ_1 of Eq. (1) are rational in the direction, which is itself a projective function of the pixel, so the exact residual over a neigh-

bourhood is expensive to obtain. A bounding rectangle does not need it, only how fast the response falls off around its peak, so we expand about $\mathbf{x}^{\mathrm{ref}}$, where that peak sits, and take

$$\mathbf{J}_i = \frac{\partial \mathbf{r}_i}{\partial (x, y)} \in \mathbb{R}^{2\times 2}, \tag{11}$$

the rate at which the residual moves per pixel step, evaluated by finite differences from Eq. (1) at $\mathbf{x}^{\mathrm{ref}}$ and at $\mathbf{x}^{\mathrm{ref}}$ displaced by one pixel along each axis. Those coordinates need not be integers: Eq. (1) is defined at any point of the image plane, and a unit step is only the interval over which the difference is taken. This costs three evaluations of a map that is already needed for rendering, and avoids differentiating the projective composition. Note that this linearization decides only which tiles a primitive is listed in. Once a pixel is being shaded, its residual is computed from Eq. (1) and Eq. (4) directly, so a listed primitive contributes exactly what Eq. (5) says, and a slightly loose rectangle costs only a few wasted evaluations.

From the residual to a screen extent. The rasterizer needs a range of pixels, while Eq. (11) gives the rate at which the residual grows as the pixel moves away from $\mathbf{x}^{\mathrm{ref}}$. What connects them is the point at which the response becomes negligible: the kernel evaluates Eq. (5) only while the query stays within five standard deviations of the ray group and discards it beyond, that is, while

$$\mathbf{r}_i^\top \left(\Sigma_i + \rho_i^2 \mathbf{I}\right)^{-1} \mathbf{r}_i \le 25, \tag{12}$$

with $\Sigma_i + \rho_i^2\mathbf{I}$ the covariance the kernel reads (Sec. 3.1). A rate and a cutoff together give a range.

Let $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^{\mathrm{ref}}$ be the offset of a pixel from the reference position of primitive i. Expanding Eq. (4) about $\mathbf{x}^{\mathrm{ref}}$,

$$\mathbf{r}_i(\mathbf{x}) = \underbrace{\mathbf{r}_i(\mathbf{x}^{\mathrm{ref}})}_{=\mathbf{0}} + \mathbf{J}_i\,\Delta\mathbf{x} + O(\|\Delta\mathbf{x}\|^2), \tag{13}$$

where the leading term drops out because $\mathbf{x}^{\mathrm{ref}}$ is where the residual vanishes. Substituting the first-order part into Eq. (12), the pixels that matter are those with

$$\Delta\mathbf{x}^\top \mathbf{M}_i\,\Delta\mathbf{x} \le 25, \qquad \mathbf{M}_i = \mathbf{J}_i^\top \left(\Sigma_i + \rho_i^2\mathbf{I}\right)^{-1}\mathbf{J}_i. \tag{14}$$

Read as a quadratic form, $\mathbf{M}_i$ takes an offset in pixels and returns the squared Mahalanobis distance of the query at that pixel from the ray group. It is therefore an inverse covariance in pixel coordinates: a primitive whose residual moves quickly per pixel step crosses the cutoff within a few pixels, while one whose residual barely moves covers many. The kernel already holds the Cholesky factor $\mathbf{L}_i$ of $\Sigma_i + \rho_i^2\mathbf{I}$, so $\mathbf{M}_i$ is formed as $(\mathbf{L}_i^{-1}\mathbf{J}_i)^\top(\mathbf{L}_i^{-1}\mathbf{J}_i)$ and no inverse is taken in the implementation.

The pixels satisfying Eq. (14) form an ellipse centred on $\mathbf{x}^{\mathrm{ref}}$, since $\mathbf{M}_i$ is positive definite. Tiles are axis aligned, so what the rasterizer needs from that ellipse is its bounding rectangle, that is, the largest $|\Delta x|$ and $|\Delta y|$ it contains. For an ellipse $\Delta\mathbf{x}^\top\mathbf{M}\Delta\mathbf{x} \le c^2$, the extreme value of a coordinate $\mathbf{a}^\top\Delta\mathbf{x}$ is $c\sqrt{\mathbf{a}^\top\mathbf{M}^{-1}\mathbf{a}}$, so taking $\mathbf{a} = (1,0)^\top$ and $(0,1)^\top$ gives the two half-widths, written ext_x and ext_y, as $c\sqrt{(\mathbf{M}^{-1})_{11}}$ and $c\sqrt{(\mathbf{M}^{-1})_{22}}$. Writing

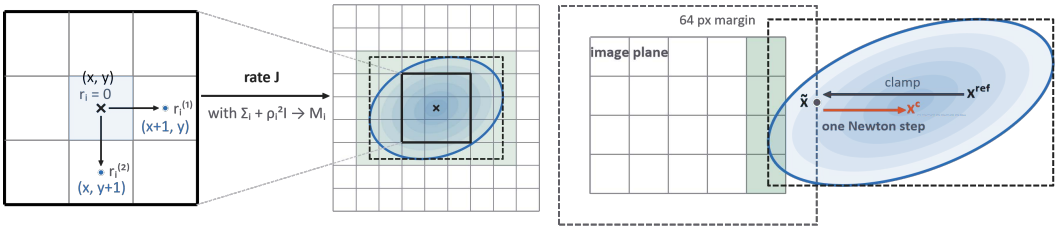
$$\mathbf{M}_i = \begin{pmatrix} A & B \\ B & C \end{pmatrix}, \quad \mathbf{M}_i^{-1} = \frac{1}{AC - B^2}\begin{pmatrix} C & -B \\ -B & A \end{pmatrix}, \quad \mathrm{ext}_x = \sqrt{\frac{25\,C}{AC-B^2}}, \quad \mathrm{ext}_y = \sqrt{\frac{25\,A}{AC-B^2}}, \tag{15}$$

with $c = 5$ from Eq. (12), so no eigendecomposition is needed. Both are clamped to $[3, 1024]$ pixels: the lower bound keeps a primitive listed in at least the tile it peaks in, and the upper bound caps how many entries one primitive can emit when $AC - B^2$ is small, the near-singular case treated next. Every tile the rectangle overlaps receives one entry for i. We plot the tile assignment in Fig. 6 (a).

When the reference position falls far outside the image. Write $\mathbf{P}_i = \xi_i\mathbf{e}_1 + \eta_i\mathbf{e}_2 + \mathbf{n}/p_i$ for the point on the plane $z = 1/p_i$ the group is defined by. The query that belongs to the group leaves the camera centre $\mathbf{o}_j$ towards $\mathbf{P}_i$, so its direction is $\mathbf{d} = \mathbf{P}_i - \mathbf{o}_j$, and writing $\mathbf{m} = KR^{-1}\mathbf{d} = (m_1, m_2, m_3)^\top$, which by Eq. (1) is proportional to $(q_x, q_y, 1)^\top$,

$$\mathbf{x}^{\mathrm{ref}} = \left(\frac{m_1}{m_3}, \frac{m_2}{m_3}\right), \tag{16}$$

with m_3 the depth of $\mathbf{P}_i$ in the camera frame, the same quantity used as the sorting key. Primitives with m_3 below 0.05 are culled before this point, but a small positive m_3 still sends Eq. (16) far



(a) From residual evaluations to the tiles (b) Reference position falls off-image

Figure 6: **Tile assignment without projection.** Each cell is a pixel. Green marks the pixels a primitive can affect. **(a)** Three evaluations of Eq. (1), one pixel apart, give the rate $\mathbf{J}_i$ by Eq. (11), which together with $\Sigma_i + \rho_i^2\mathbf{I}$ forms the falloff $\mathbf{M}_i$ of Eq. (14); the blue ellipse encloses the pixels satisfying $\Delta\mathbf{x}^\top\mathbf{M}_i\Delta\mathbf{x} \le 25$, and the dashed rectangle bounds it. **(b)** When $\mathbf{x}^{\text{ref}}$ falls beyond the image margin it is clamped to $\tilde{\mathbf{x}}$, and one Newton step re-centres the rectangle at $\mathbf{x}^{\text{c}}$, the zero of the first-order model.The rectangle still reaches the image, so the primitive is kept and assigned to the tiles the rectangle covers.

outside the image, large enough to overflow the tile arithmetic that follows. We therefore clamp $\mathbf{x}^{\text{ref}} = (x, y)$ componentwise to a rectangle 64 pixels wider than the image on every side,

$$\tilde{\mathbf{x}} = \big(\min(\max(x, -64),\, W + 64),\ \min(\max(y, -64),\, H + 64)\big), \tag{17}$$

for an image of width W and height H. Each coordinate that already lies in its range is left alone, and one that does not is replaced by the nearest endpoint, so a value of $(300, 50)$ on a 100×100 image becomes $(164, 50)$. This keeps the arithmetic finite, but it moves the centre off the zero of the residual, where Eq. (13) was expanded.

One step of Newton's method puts it back. At $\tilde{\mathbf{x}}$ the residual $\mathbf{r}_i(\tilde{\mathbf{x}})$ is nonzero, and the first-order model $\mathbf{r}_i(\mathbf{x}) \approx \mathbf{r}_i(\tilde{\mathbf{x}}) + \mathbf{J}_i(\mathbf{x} - \tilde{\mathbf{x}})$ vanishes at

$$\mathbf{x}^{\text{c}} = \tilde{\mathbf{x}} - \mathbf{J}_i^{-1}\,\mathbf{r}_i(\tilde{\mathbf{x}}). \tag{18}$$

Both quantities on the right are already at hand: the three calls to Eq. (1) behind Eq. (11) are taken at $\tilde{\mathbf{x}}$ rather than at $\mathbf{x}^{\text{ref}}$, which may be too large to difference safely, so the same $\mathbf{J}_i$ serves the rectangle and this step. When no clamping occurred the residual at $\tilde{\mathbf{x}}$ is zero and $\mathbf{x}^{\text{c}}$ returns $\mathbf{x}^{\text{ref}}$ unchanged; otherwise it recovers where the linearization places the zero. The rectangle is centred at $\mathbf{x}^{\text{c}}$, so a primitive whose reference position lies outside the image is still listed in the tiles its falloff reaches. Fig. 6 (b) plots this case.

Every primitive past the near cull follows these same three steps, whether or not its reference position lies in the image: the three evaluations behind $\mathbf{J}_i$, the solve of Eq. (18), and the check of the rectangle against the image. Nothing cheaper can be checked first, because whether a primitive reaches the image depends on its extent, and the extent comes from $\mathbf{J}_i$. The clamp of Eq. (17) is two comparisons per coordinate and leaves any primitive whose reference position is already within the margin untouched.

The step is rejected in two cases. A step longer than 128 pixels is outside the range over which the linearization was intended to hold. And $\mathbf{J}_i$ may be near singular, the grazing configuration: the view moves the residual in essentially one direction, the response is nearly constant along a line in the image, and no single zero exists to move towards. In both cases $\tilde{\mathbf{x}}$ serves as the centre and both half-widths are raised to at least 32 pixels, so the primitive is listed over a band rather than at a point.

Ordering and culling. The depth key is m_3, the depth of $\mathbf{P}_i$ in the camera frame, which is the ordering rule of Sec. 3.1. Primitives with m_3 below 0.05, or whose rectangle does not meet the image, are dropped. With bounds and keys in hand the remainder is standard: each primitive emits one entry per tile its rectangle overlaps, the entries are sorted by tile and then by depth, and each tile composites its own list with Eq. (7). The backward pass replays that traversal in reverse. Only three steps differ from a point-based rasterizer, the reference position, the rate $\mathbf{J}_i$, and the Newton re-centring, and none of them requires a projected covariance.

D ANTI-ALIASING AND DEFOCUS DEBLURRING

A pixel does not correspond to a single ray but a small bundle of them, and both effects treated here stem from how that bundle spreads: aliasing appears when the pixel grid is too coarse for its spatial spread, and defocus when a finite aperture spreads it over directions. LFP accounts for the bundle through a term it already carries. Sec. 3.1 renders with $\Sigma_i + \rho_i^2\mathbf{I}$ so that no response is read more narrowly than a view can resolve, and this same term admits a second reading. Let the queries of a pixel's bundle differ from its central ray by $\boldsymbol{\delta}$, with mean zero and covariance P in residual space. Since the response Eq. (5) is an unnormalized Gaussian and Gaussians are closed under convolution, with covariances adding (Yu et al., 2024a),

$$\mathbb{E}_{\boldsymbol{\delta}}\big[G_i(\mathbf{r}_i - \boldsymbol{\delta})\big] = \sqrt{\frac{\det \Sigma_i}{\det(\Sigma_i + P)}}\ \exp\Big(-\tfrac{1}{2}\mathbf{r}_i^\top (\Sigma_i + P)^{-1}\mathbf{r}_i\Big). \tag{19}$$

Two things happen and no more: the covariance gains P, and the peak falls by the square root of a determinant ratio. The first is why each effect below is a term added to Σ_i, with the record, the response, and the compositing left untouched; the second is a change in how much a primitive contributes, handled differently in the two settings for the reason given in Sec. D.3. Classical light field rendering obtained both effects in just this way, by widening what a stored entry averages over (Levoy & Hanrahan, 1996; Isaksen et al., 2000).

The pixel's own bundle is the case $P = \rho_i^2\mathbf{I}$. The value of ρ_i follows from the units of Eq. (4): a pixel of unit width subtends a half-angle $1/(2f)$, which at a primitive of camera depth Z_{ij}^{cam} is a transverse displacement $Z_{ij}^{\mathrm{cam}}/(2f)$, and Eq. (4) measures that displacement scaled by p_i, so $\rho_i = Z_{ij}^{\mathrm{cam}}\, p_i/(2f)$, with $f = \max(f_x, f_y)$ when the two differ. Taking the half-pixel as one standard deviation is a convention rather than a consequence; what matters below is only the ratio t_i/ρ_i^2, which is the same under any convention.

D.1 ANTI-ALIASING

A coarser output enlarges the spatial spread of the bundle, so we render with $\Sigma_i + t_i\mathbf{I}$. The peak factor of Eq. (19) is a change in the total a primitive contributes, since an unnormalized Gaussian integrates to $2\pi\sqrt{\det\Sigma}$ and preserving that total under a covariance change costs a factor $\sqrt{\det\Sigma/\det\Sigma'}$ on the opacity (Darmon et al., 2024). During training it is never applied explicitly: o_i is free and is fitted with ρ_i^2 in place, so the converged o_i already contains it. Rendering at another scale replaces ρ_i^2 by t_i, and what must be applied on top of the fitted opacity is the ratio of the two peak factors, in which $\det\Sigma_i$ cancels,

$$\alpha_i \ \leftarrow\ \alpha_i \sqrt{\det(\Sigma_i + \rho_i^2\mathbf{I})/\det(\Sigma_i + t_i\mathbf{I})}, \tag{20}$$

which is the correction Yu et al. (2024a) apply to their own low-pass filter for the same reason. The cancellation makes it well defined without knowing the fitted o_i, and makes it exactly one when $t_i = \rho_i^2$, so rendering at the training scale reproduces ordinary rendering.

Calibrating t_i takes no free parameter. At $1/s$ of the training resolution an output pixel averages an $s \times s$ block of training pixels, the box filter of the imaging process that Yu et al. (2024a) also model, whose variance is $(s^2-1)/12$ per axis, and each of those pixels carries its own footprint of variance $1/4$ in the convention above. The two are independent, so

$$t_i = \rho_i^2\Big(1 + \frac{s^2-1}{3}\Big), \tag{21}$$

which is one at $s = 1$ and 22 rather than 64 at $s = 8$, the difference being the part of the widening the training footprint already supplied. Two details matter for reproduction: both determinants are evaluated on the Cholesky factors the rasterizer draws with, which are floored for numerical safety, since taking them from the unfloored parameters over-corrects elongated primitives; and primitives culled by the near plane are given $\rho_i = 0$ on both sides.

At an output finer than the training views the same construction would give $t_i < \rho_i^2$, narrowing every primitive below the width it was fitted at. Since no observation supports a footprint narrower than the training one, we hold t_i at ρ_i^2 there, which leaves Eq. (20) at one. Either way a trained scene renders at any resolution with no re-optimization.

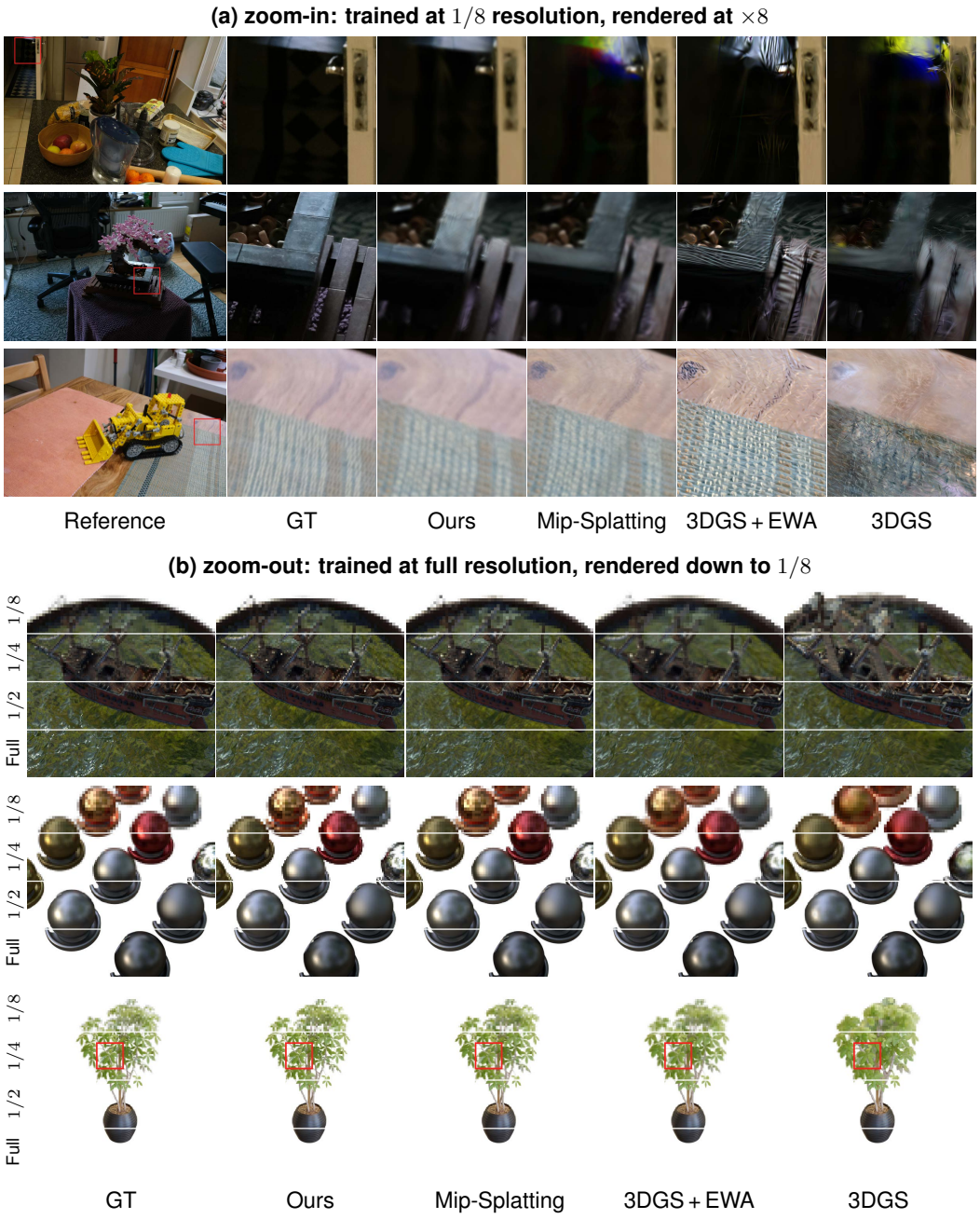


Figure 7: Qualitative examples of anti-aliasing under single-scale training, multi-scale testing.

Besides the quantitative results presented in Tab. 2, we also present the qualitative results in Fig. 7 to show the effectiveness of LFP for the anti-aliasing task. We notice that despite its simplicity, LFP performs favorably against Mip-splatting in both the zoom-in and zoom-out settings. In Fig. 7 (a), where the models are trained at $1/8$ resolution on mip-NeRF 360 and rendered at $\times 8$, the reference view marks each crop. 3DGS breaks into needle artifacts, 3DGS+EWA hallucinates scratch-like structures, and Mip-Splatting smears colour and washes out fine texture, while ours stays faithful to the ground truth. In Fig. 7 (b), all models are trained at full resolution on Blender and rendered down to $1/8$; each image is split into horizontal bands rendered at the resolutions marked on the left ($1/8$ on top), low-resolution bands shown with their actual pixels. 3DGS brightens and dissolves thin structures at low resolutions, while ours stays close to the ground truth at every rate, similar to the specially designed Mip-Splatting.

D.2 Defocus and Refocus

A finite aperture spreads the bundle over directions rather than over the image: the queries of a pixel leave from across the aperture instead of from one center. A thin lens of aperture K_j focused at F_j spreads them over the circle of confusion, of angular radius

$$\sigma_{ij} = \frac{K_j}{2}\Big|\frac{1}{F_j} - \frac{1}{Z_{ij}}\Big|, \tag{22}$$

the standard thin-lens quantity also used by Wu et al. (2022); Wang et al. (2025), with Z_{ij} the depth already computed for compositing (Sec. 3.1). It vanishes at $Z_{ij} = F_j$, so a primitive on the focal plane is unaffected at any aperture, and it is linear in K_j.

To reach P the spread is carried into residual space. Both Eq. (1) and Eq. (4) are explicit, the first rational in the direction and the second affine in the coordinates, so their composition has a first-order derivative in closed form: about the central ray of the pixel, $\mathbf{Q}_{ij} = \partial \mathbf{r}_i / \partial \mathbf{q} \in \mathbb{R}^{2\times 2}$ is fixed for primitive i in view j, with $\mathbf{q} \in \mathbb{R}^2$ parameterizing the direction on the normalized image plane. A circular aperture spreads $\mathbf{q}$ isotropically, so $P = \sigma_{ij}^2 \mathbf{Q}_{ij}\mathbf{Q}_{ij}^\top$ and Eq. (19) gives

$$\Sigma_i \;\rightarrow\; \Sigma_i + \sigma_{ij}^2\, \mathbf{Q}_{ij}\mathbf{Q}_{ij}^\top. \tag{23}$$

The only quantities added are K_j and F_j, two scalars per training image optimized with the scene; $\mathbf{Q}_{ij}$ follows from the parameterization and is not learned. Two approximations enter and no others: the spread is carried to first order through the constant $\mathbf{Q}_{ij}$, and the disc is replaced by a Gaussian of matching second moment so that Eq. (19) applies. Neither the response Eq. (5) nor the compositing is approximated.

Applying Eq. (23) when rendering the training views, whose images are defocused, and dropping it at test time forces a scene that renders sharp to account for blurred observations, which is defocus deblurring. And since K_j and F_j describe the camera rather than the scene, they stay free once training ends: sweeping F_j moves the focus through a trained scene and raising K_j renders a depth of field at apertures never captured, recovering the synthetic aperture of classical light field rendering (Isaksen et al., 2000) without its dense table.

In a representation anchored in 3D, no coordinate records where within the aperture a query passes, so the aperture has to be stated in the operator instead: an analytic circle of confusion attached to each projected footprint (Wang et al., 2025), a network predicting a per-primitive deformation (Lee et al., 2024b), or blur kernels fitted per view (Peng et al., 2024). Here it lands on the covariance the renderer already reads, beside the term for the pixel.

D.3 The peak factor at training and at test time

The peak factor of Eq. (19) is treated differently in the two settings, and the reason is which quantities are still free. While fitting a scene to defocused images, o_i and the per-image K_j, F_j are optimized through the same loss and settle at whatever reproduces the observations with the factor absent, so we do not apply it. Supplying it explicitly does not help: retraining with the factor raised to a range of powers was worse on every metric at every power, with a monotone return toward the released behaviour as the power approaches zero, which is what one expects if the factor is already absorbed.

Refocusing a trained scene is the opposite case. There o_i is frozen at a value fitted under the training footprint and a new (F, K) is imposed that no training image carried, so the factor is a real change to the render and is applied exactly as in Eq. (20), with the aperture term in place of the scale term. Because Eq. (23) approximates a disc by a Gaussian inside an alpha composite, the result is not exact: measured against a reference that samples the aperture directly, the closed form is a few percent darker over a frame, and a thin bright feature against a dark background loses about a third of its brightness, since such a feature is carried by few primitives whose footprints grow fastest relative to their original size.

E Ablation Study

All ablations are run on the 9 scenes from Barron et al. (2022) under the setting of Tab. 1.

View-dependent term. Tab. 7 replaces the three ASG lobes with the spherical harmonics of 3DGS (Kerbl et al., 2023) and with a constant colour, and repeats the three on 3DGS itself. Both

Table 7: Directional appearance model comparisons, i.e., SH versus ASG, on the 9 scenes of Mip-NeRF 360. The last column counts the floats each primitive spends on appearance. The same three models are attached to 3DGS for reference.

	Appearance	PSNR ↑	SSIM ↑	LPIPS ↓	Floats
3DGS	RGB only	26.82	0.804	0.268	3
	SH	27.46	0.815	0.255	48
	ASG	27.69	0.816	0.255	30
LFP	RGB only	26.76	0.798	0.264	3
	SH	27.57	0.808	0.256	48
	ASG	28.02	0.820	0.247	30

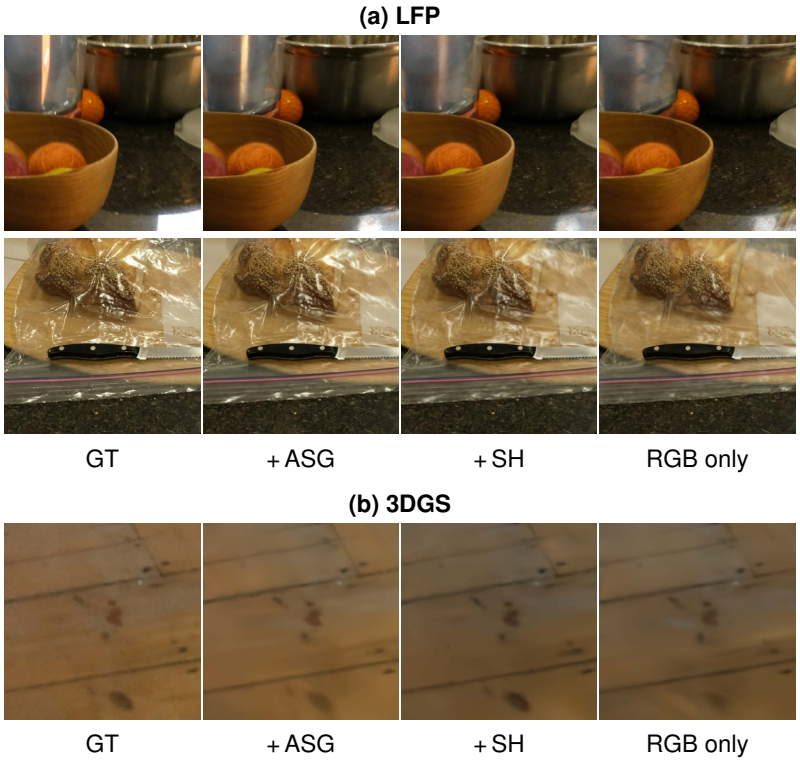


Figure 8: Appearance-model ablation. The view-dependent appearance model is replaced by spherical harmonics (SH, degree 3) or by a view-independent colour (RGB only), under an otherwise identical training protocol: (a) inside LFP, (b) inside 3DGS.

sides use the SH basis and constants of 3DGS, each evaluated in the direction that representation provides: for 3DGS the direction from the camera to a primitive, constant over its extent on screen, and for LFP the direction of the query ray, which the parameterization carries per pixel. Within each side the two bases share that direction, so the comparison is controlled. The lobes differ in one further respect: an LFP primitive carries a direction of its own, that of its anchor ray (Sec. 3.2), so its lobes are evaluated in the reflection about it, while a 3D Gaussian carries none and its lobes stay in the same viewing direction its harmonics use. Lobe axes are learned on both sides, and the appearance budget is matched at 30 floats per primitive.

With a constant colour the two representations are level, so whatever difference appears later comes from the directional term alone. On 3DGS the choice hardly matters, lobes and harmonics being close: with only the viewing direction to work with, the lobes act as one more basis over directions. On LFP they are ahead by roughly twice the margin with fewer floats, because they can reflect. A ray group collects rays with coherent radiance, so the base carries what holds across the group, and what the lobes are left with is the highlights, which is what a reflection about a normal describes.

Fig. 8 (a) isolates the effect inside our representation. With a view-independent colour (RGB only), a primitive has to explain all of its observations with a single value, so specular energy is averaged away: the reflections on the steel pot and the specular wrinkles of the plastic bag flatten into a matte surface, and the structure they reveal – the rim of the pot, the serrated blade behind the bag – loses its contrast. Degree-3 SH brings part of that energy back, but only at low angular frequency: the highlights return as broad, smeared glows whose position and extent do not follow the ground truth, while the fine creases of the bag stay washed out. This is not a matter of capacity: at degree 3 the SH block carries 45 coefficients per primitive against the 27 of our three ASG lobes. What differs is where those coefficients can be spent – SH spreads them isotropically over the whole sphere, whereas a single ASG lobe concentrates them on the narrow, anisotropic angular support that a highlight actually occupies. Our full model reproduces both the intensity and the shape of these reflections. Fig. 8 (b) repeats the same substitution inside 3DGS, replacing its spherical harmonics by our ASG lobes and leaving the rest of the pipeline untouched. The trend is unchanged: SH renders the sheen of the wooden floor pale and flat, while ASG recovers its warm tone and the grain that the sheen reveals. The gain therefore comes from the angular basis itself rather than from our primitive, so ASG can be dropped into existing Gaussian splatting appearance models, and its advantage is largest wherever radiance varies sharply with view direction.

Different lobes for ASG. The number of ASG lobes K is a compile-time constant of the rasteriser: every per-pixel lobe loop and per-thread cache is sized by it, so changing K means rebuilding the kernel rather than flipping a flag. We retrain all nine Mip-NeRF 360 scenes for $K \in \{1, 3, 5\}$, holding the atlas count, initialisation and schedule fixed, and keeping one out-of-sigmoid lobe whenever $K > 1$. Quality peaks at $K = 3$ (28.02 dB PSNR, 0.820 SSIM, 0.247 LPIPS): dropping to a single lobe costs 0.15 dB (27.87 dB) while raising K to five does not recover it (27.97 dB), because the extra lobes add per-primitive work without adding usable angular detail – the optimiser in fact settles on 3% fewer primitives at $K=5$ than at $K=3$. Rendering cost, in contrast, is monotone in K: 120.6 FPS at $K=1$, 101 FPS at $K=3$ and 84.4 FPS at $K=5$. $K=3$ is therefore the knee of the quality–speed curve (Fig. 9) and is what we ship; $K=1$ remains a useful operating point when throughput matters more than the last 0.15 dB.

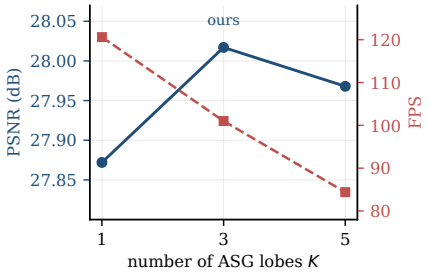


Figure 9: Lobe count K: quality peaks at 3, cost grows monotonically.

Highlight regularization. Removing $\lambda_g \|\mathbf{g}\|^2$ from Eq. (9) costs **0.17** dB (28.02 → 27.85 averaged over the nine Mip-NeRF 360 scenes, with eight of the nine degrading), together with 0.005 SSIM and 0.008 LPIPS. Without it the lobes are free to reproduce radiance that the base $\mathbf{c}_i$ should carry, since a highlight centred near the training directions is cheaper to fit than a correction to the base, and the split between the two is then set by initialization rather than by the data. Tab. 9 shows the effect: **the mean penalized lobe amplitude grows** $6.6\times$ **(0.098 → 0.640) while the base colour is left essentially untouched** $(1.03\times)$**, i.e. the surplus radiance is absorbed by the lobes rather than by** $\mathbf{c}_i$. The model also ends up **12% smaller** (2.00 M → 1.75 M primitives), because the lobes soak up photometric error that would otherwise drive densification.

Atlas design. A single plane pair cannot index rays that run nearly parallel to it (Sec. 3.2), so the number of atlases sets how much of the ray space is reachable. Tab. 8 varies that number. With one atlas the reconstruction loses 6.92 dB, and the loss is of the same order indoors and outdoors, 7.12 against 6.76 dB, so what is missing is coverage rather than anything particular to a capture pattern. Four atlases recover 6.62 of those 6.92 dB, eight a further 0.18, and sixteen another 0.12, while each doubling raises the per-camera precomputation and places more primitives on the camera-frame ordering of Sec. 3.1. Thirty-two adds 0.01 dB and is behind sixteen outdoors, so coverage is saturated by then. We use sixteen, the smallest count on that plateau.

Letting the atlas normals be optimized rather than fixed before training never helps, and at the largest step size we tried it costs 0.18 dB. A moving normal changes the coordinates of every query it serves, so the residuals of all primitives assigned to that atlas shift together, and the gradient it receives averages over them rather than correcting any one. The measurements show this cancellation. At 10^{-5} and 10^{-4} the normals barely rotate, $0.10°$ and $0.28°$ after 30k iterations, and every metric is unchanged to within noise. Only at 10^{-3} do they move appreciably, $0.60°$ on average and $6.3°$ at

Table 8: Atlas design, each metric reported over all scenes and split into the five outdoor and four indoor ones. Top: number of atlases, normals fixed at the cluster centres of the training camera directions. Bottom: the same 16 atlases with the normals left free, at three step sizes, larger learning rate (lr) means more movement for the intial normals.

	PSNR ↑			SSIM ↑			LPIPS ↓		
	all	outdoor	indoor	all	outdoor	indoor	all	outdoor	indoor
Number of atlases									
1	21.10	17.95	25.03	0.657	0.512	0.838	0.393	0.460	0.310
4	27.72	24.34	31.95	0.811	0.718	0.929	0.257	0.276	0.234
8	27.90	24.64	31.98	0.817	0.728	0.929	0.251	0.264	0.234
16 (ours)	28.02	24.71	32.15	0.820	0.731	0.930	0.247	0.259	0.232
32	28.03	24.70	32.19	0.819	0.730	0.931	0.247	0.260	0.231
16 atlases, normals learned									
lr 10^{-5}	27.97	24.69	32.08	0.818	0.729	0.930	0.249	0.262	0.232
lr 10^{-4}	27.96	24.68	32.08	0.818	0.729	0.930	0.249	0.262	0.232
lr 10^{-3}	27.84	24.69	31.79	0.815	0.727	0.925	0.254	0.268	0.235

Table 9: Highlight regularization, averaged over the nine Mip-NeRF 360 scenes. $\|\mathbf{g}\|$ is the mean amplitude of the penalized lobes and $\|\mathbf{c}\|$ the mean base-colour magnitude. Dropping the term lets the lobes grow by $6.6\times$ while the base is left untouched.

	PSNR ↑	SSIM ↑	LPIPS ↓	$\|\mathbf{g}\|$	$\|\mathbf{c}\|$	#prim.
with $\lambda_g\|\mathbf{g}\|^2$ (ours)	28.02	0.820	0.247	0.098	1.79	2.00 M
without	27.85	0.815	0.255	0.640	1.85	1.75 M
Δ	-0.17	-0.005	$+0.008$	$\times 6.57$	$\times 1.03$	-12%

most, and there all three metrics degrade, the indoor scenes by 0.36 dB against 0.02 dB outdoors, which have fewer primitives per atlas and so fewer residuals to average. Fixing the normals at the cluster centres of the training camera directions leaves this degree of freedom out of the optimization, and the primitives absorb the difference through p_i and (μ_i^s, μ_i^t).

Optimizing (μ^u, μ^v). A primitive is anchored by two plane coordinates, (μ_i^u, μ_i^v) on the entry plane and (μ_i^s, μ_i^t) on the exit plane; by default we optimize only the latter and keep (μ_i^u, μ_i^v) at its initial value. Enabling gradients is not enough to change this, since the backward kernel never writes $\partial\mathcal{L}/\partial\boldsymbol{\mu}_i^{uv}$ and the gradient reaching those parameters is identically zero. We therefore add the missing term. By Eq. (4) the residual depends on the entry coordinates through the coupling coefficient, so $\partial\mathbf{r}_i/\partial\boldsymbol{\mu}_i^{uv} = +(1-p_i)\mathbf{I}$ alongside the existing $\partial\mathbf{r}_i/\partial\boldsymbol{\mu}_i^{st} = -\mathbf{I}$, and we retrain all nine scenes at the learning rate used for (μ_i^s, μ_i^t).

The result is unchanged within run-to-run noise, 28.01 dB / 0.820 / 0.247 against 28.02 dB / 0.820 / 0.247 for the anchored model, with per-scene differences spanning -0.07 to $+0.03$ dB and no consistent sign. The parameters do move. Over a 3000-step probe their cumulative displacement is 57% of that of (μ_i^s, μ_i^t), and a ten-fold learning rate moves them five times further without improving any metric, so the outcome is not an artefact of under-training.

This is what the parameterization predicts. A ray group is fixed by its disparity and any one of its rays (Sec. 2), so $\boldsymbol{\mu}_i$ carries one degree of freedom that names the group and one that only chooses which member of it is stored. Optimizing (μ_i^s, μ_i^t) already spans the first, and what freeing (μ_i^u, μ_i^v) adds is the second, which no photometric loss can see: the response Eq. (5) depends on the group and not on which of its rays the record was written against. We therefore keep (μ_i^u, μ_i^v) fixed, which removes two learnable parameters and two atomic accumulations per primitive at no cost in quality.