SCENEFOAM: SPARSE-VIEW RECONSTRUCTION OF NATIVE RADIANCE FOAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

SceneFoam learns to reconstruct native Power Foam fields from sparse calibrated images, recovering bounded cells, internal surfaces, local texture, and view-dependent color. Its central design aligns appearance updates with the geometry that determines visibility and pixel-to-texture contributions. An image-conditioned frontend and native initializer are trained jointly to predict the scene. Four source-conditioned color-fitting steps then align directional radiance with the input photographs and independent reference images. A geometry-first refiner updates the surfaces, recomputes their image residuals and source-loss gradients, and predicts appearance using this refreshed evidence. Together, these stages couple a learned scene prior with observations evaluated on the current foam. On eight-view DL3DV140, SceneFoam obtains 27.97 dB PSNR, 0.873 SSIM, and 0.153 LPIPS, outperforming ReSplat on all three metrics. On four native-PF comparison scenes, it improves all three metrics over both 30,000-update optimization baselines with complete inference in 8.94–9.44 s. The recovered field directly supports rasterization, ray tracing, distorted cameras, and interaction through its tetrahedral dual. SceneFoam brings high-quality sparse-view acquisition to textured spatial partitions through geometry-consistent evidence updates.

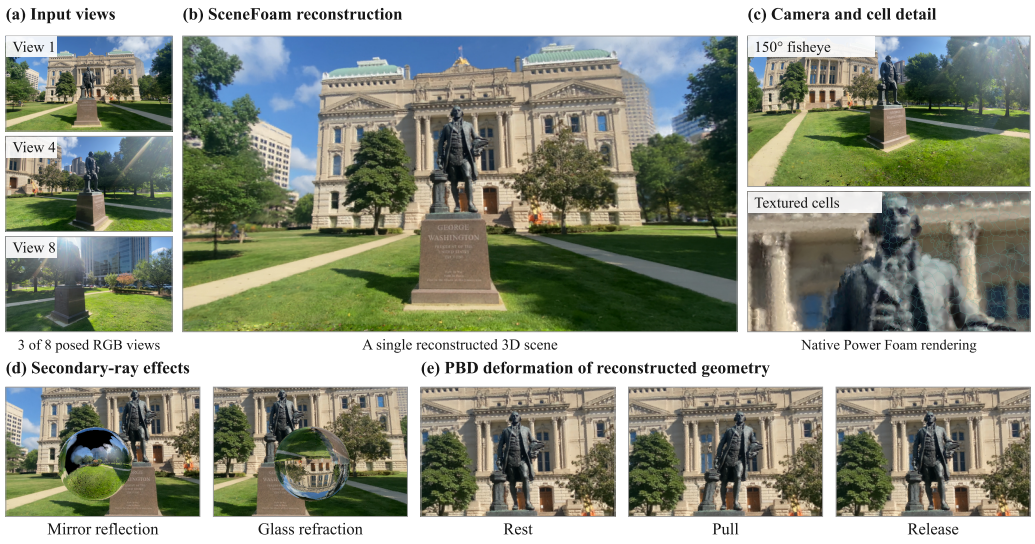

Figure 1: **Sparse views to one native foam for rendering and interaction.** (a,b) Eight posed images (three shown) yield one reconstructed scene. (c) Native fisheye rendering and textured-cell detail. (d) Mirror and glass insertions query reflected and refracted rays. (e) PBD deformation uses its extended variant, XPBD, directly on the native tetrahedral dual: rest, pull, and release. All panels use the same reconstructed scene; simulation details appear in Appendix H.5.

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.

1 INTRODUCTION

Sparse-view reconstruction turns a small set of calibrated photographs into a reusable scene. Learned methods recover continuous radiance fields, Gaussian primitives, or triangles (Yu et al., 2021; Chen et al., 2024a; Xu et al., 2026; Xiong et al., 2026). We study the acquisition of *native textured radiance foams*: spatial partitions whose bounded cells contain internal surfaces and local directional texture. This target combines image synthesis with a cell complex that supports rendering and interaction.

We introduce **SceneFoam**, a learned sparse-view reconstruction framework built on the Power Foam representation (Govindarajan et al., 2026). Its predicted parameters directly serve native rasterization and ray tracing, including distorted-camera and secondary-ray queries. The cell centers also provide mechanical nodes on the native tetrahedral dual (Figure 1). SceneFoam learns to acquire the geometry and radiance that make these interfaces useful from sparse observations.

The central design follows the geometry dependence of image evidence. Cell boundaries control spatial support, internal surfaces determine occupied ray segments, and surface-attached texture sites interpolate directional color. A geometry update changes visibility and the contribution of each texture site to each pixel. Appearance therefore benefits from measurements evaluated on the *updated* geometry. This dependence gives a concrete structure to learned foam reconstruction: condition radiance on the current surfaces, update geometry, and refresh texture-site evidence before predicting appearance.

SceneFoam implements this structure through jointly learned prediction, source-conditioned color fitting, and geometry-first refinement. The image frontend and foam initializer learn together through native rendered-image supervision. Four color-fitting steps use real photographs and independent references with geometry fixed. The refiner then updates geometry, re-renders source views, and lifts residuals and source-loss gradients to the revised texture support. Its appearance head receives these refreshed measurements together with the predicted geometry update.

SceneFoam improves all three image-quality metrics over ReSplat on eight-view DL3DV140, two-view RealEstate10K, and transfer to ACID. Against native PF optimization from original or learned initialization with virtual references, it improves every metric in all four comparison scenes while completing reconstruction in 8.94–9.44 s. Controlled ablations compare simultaneous updates, sequential updates with retained evidence, and geometry-first evidence refresh.

Contributions. We develop learned sparse-view acquisition of native textured Power Foam fields; introduce source-conditioned radiance fitting and geometry-consistent texture-evidence refresh within the acquisition process; and demonstrate high-quality reconstruction, rapid native PF acquisition, and reuse of the recovered field across rendering and interaction tasks.

2 RELATED WORK

Image-conditioned scene reconstruction. pixelNeRF, IBRNet, and MVSNeRF condition radiance fields on image features, multi-view aggregation, and stereo cost volumes (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021). For Gaussian prediction, pixelSplat estimates image-pair depth distributions (Charatan et al., 2024); MVSplat and MVSGaussian exploit multi-view matching (Chen et al., 2024a; Liu et al., 2024); DepthSplat combines monocular and multi-view depth features (Xu et al., 2025). GS-LRM uses transformer-based reconstruction, Long-LRM handles longer image sequences, and Splatt3R reconstructs from uncalibrated pairs (Zhang et al., 2024; Chen et al., 2025b; Smart et al., 2024). SceneFoam learns a native cell-and-texture field with a frontend jointly trained through foam rendering.

Learned scene updates and rendering feedback. DeepView learns gradient-based updates to multiplane images, and G3R uses rendering gradients to update neural Gaussians (Flynn et al., 2019; Chen et al., 2024c). ReSplat uses rendering errors for recurrent Gaussian updates; iSplat also develops iterative Gaussian refinement (Xu et al., 2026; Wu et al., 2026). SplatFormer refines Gaussian scenes with a point transformer, while QuickSplat combines learned initialization and gradient-guided updates for surface reconstruction (Chen et al., 2025a; Liu et al., 2025b). SceneFoam specializes the update mechanism to surface-attached foam texture: geometry changes first, residuals and gradients

are recomputed on the new support, and appearance is decoded from that evidence within the same refinement block.

Representations and native rendering. 2DGS uses oriented planar Gaussians, and FTSplat learns triangle reconstruction (Huang et al., 2024; Xiong et al., 2026). 3DGRT traces Gaussian particles; 3DGUT combines unscented projection with ray-based particle responses; UTrice studies triangle-based rasterization and tracing (Moënne-Loccoz et al., 2024; Wu et al., 2025; Liu et al., 2025a). Radiant Foam introduces differentiable spatial-partition traversal (Govindarajan et al., 2025). Power Foam supplies bounded cells, internal surfaces, local texture, and unified rendering with spherical Voronoi directional functions (Govindarajan et al., 2026; Di Sario et al., 2026). SceneFoam contributes their learned sparse-view acquisition, organized around geometry-dependent radiance fitting and evidence refresh. Appendix I develops these connections.

3 Radiance Foams and Geometry-Conditioned Image Formation

A radiance field describes density and view-dependent color in three-dimensional space. We use Power Foam (PF) (Govindarajan et al., 2026), which stores this information in bounded cells (Figure 2). Its parameters $\mathcal{P} = (g, a, C)$ comprise geometry and spatial interpolation g, directional basis parameters a, and RGB color coefficients C. The distinction matters because fixing g and a yields a structured subproblem in C.

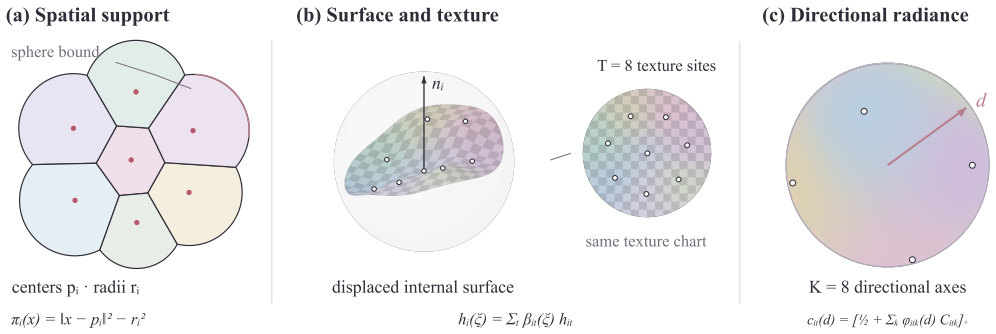

Figure 2: **The three scales of a textured radiance foam.** Bounded power cells define spatial support, surface-attached sites encode displacement and texture, and directional axes interpolate radiance. SceneFoam reconstructs these native Power Foam parameters (Govindarajan et al., 2026) from sparse images. The panels are analytical schematics.

Native geometry and texture. For cell center $p_i \in \mathbb{R}^3$ and radius $r_i > 0$, the squared-distance function π_i defines the cell region $\mathcal{V}_i$:

$$\pi_i(x) = \|x - p_i\|^2 - r_i^2, \qquad \mathcal{V}_i = \{x : \pi_i(x) \le 0,\ \pi_i(x) \le \pi_j(x)\ \forall j\}. \tag{1}$$

Each cell is the part of its bounding sphere assigned to it by these distance comparisons. An oriented surface separates a side of density σ_i from an empty side. Texture sites are sample locations on this surface that store displacement and view-dependent color. Rendering interpolates these values and combines the contributions encountered along each ray. Cell geometry therefore determines visibility and the contribution of each texture site to each pixel. Appendix A.1 gives the rendering equations.

Conditional image formation. For camera Π, fix geometry g, directional basis a, and the set of color channels clipped to zero. The rendered image is then affine—a linear function plus an offset—in the color coefficients:

$$\mathcal{R}_\Pi(g, a, C) = A_\Pi(g, a)C + b_\Pi(g, a). \tag{2}$$

The operator A_Π collects visibility, attenuation along the ray, and spatial and directional interpolation weights. Changing geometry changes these weights. We therefore fit C with geometry fixed and recompute source evidence after updating geometry. Each color update is checked against the rendered-image objective.

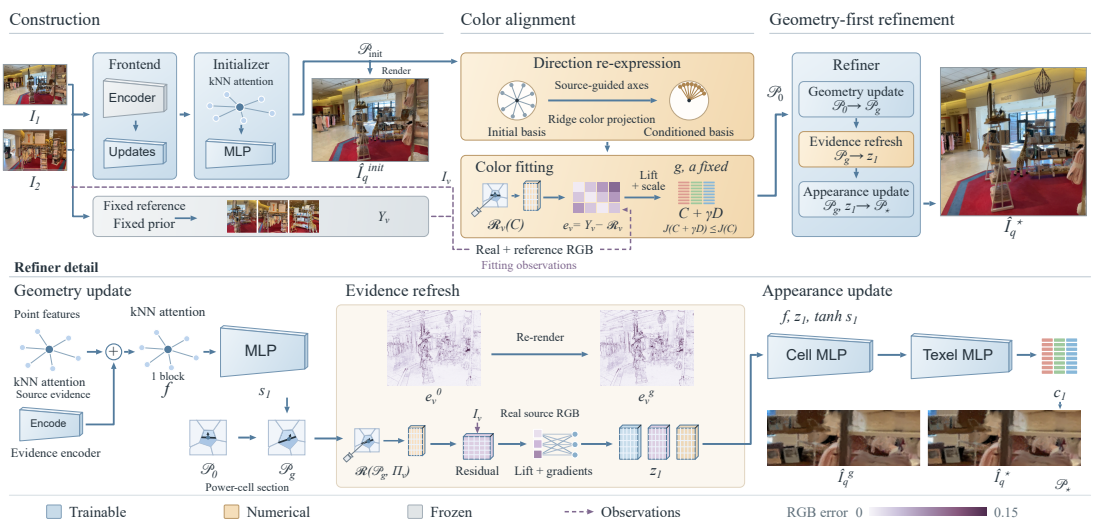

Figure 3: **Scene construction and refinement.** The lower row details the refinement network: shared features f predict geometry updates s_1; the updated scene $\mathcal{P}_g$ supplies new evidence z_1 for appearance updates c_1. MLP denotes multilayer perceptron; KNN denotes k-nearest-neighbor feature aggregation. Blue, apricot, and gray indicate learned, numerical, and fixed modules. Image panels show renders; error panels share a color scale. Cell and direction diagrams are schematic.

4 Geometry-conditioned foam reconstruction

Following the calibrated sparse-view setting of ReSplat (Xu et al., 2026), we take source photographs $\mathcal{S} = \{(I_v, \Pi_v)\}_{v=1}^{V}$ with known camera poses and intrinsics, retaining their acquisition order along the camera trajectory for virtual-view interpolation. SceneFoam constructs $\mathcal{P}_\star = (g_\star, a_\star, C_\star)$. A new camera Π_q produces $\hat{I}_q = \mathcal{R}_{\mathrm{PF}}(\mathcal{P}_\star, \Pi_q)$. Figure 3 shows the three-stage pipeline and its intermediate scenes:

$$\mathcal{S} \xrightarrow{\text{joint prediction}} \mathcal{P}_{\mathrm{init}} \xrightarrow{\text{radiance fitting}} \mathcal{P}_0 \xrightarrow{\text{geometry update}} \mathcal{P}_g \xrightarrow{\text{refreshed appearance}} \mathcal{P}_\star. \tag{3}$$

The predictor provides $\mathcal{P}_{\mathrm{init}}$; color fitting produces $\mathcal{P}_0$; the refinement network updates geometry to $\mathcal{P}_g$ and then appearance to $\mathcal{P}_\star$. Network weights stay fixed during construction. The eight-view model uses 245,760 cells, eight texture sites per cell, and eight directional RGB coefficients per site.

4.1 Jointly learned native prediction

From source images to a foam. The frontend is an image-conditioned reconstruction network initialized from ReSplat and jointly trained with the native initializer. Four recurrent updates produce Gaussian geometry G_{adapt} and spatial features H. Gaussian centers initialize cell centers, neighborhood distances determine radii and densities, and depth or covariance normals initialize surface orientations. Colors sampled from source images initialize texture, giving a seed scene $\mathcal{P}_{\mathrm{seed}}$. The initializer renders this scene at the source cameras and computes residuals, the differences between source and rendered images. It maps these residuals to texture sites using each site's contribution to each pixel. The weighted residuals, summed contribution weights, H, and current scene parameters feed spatial feature blocks and cell/site prediction heads. Their bounded outputs update geometry and color to produce $\mathcal{P}_{\mathrm{init}}$. Appendix B.1 gives the network and initialization details.

Joint training through the foam renderer. The frontend and initializer are trained jointly with a target-view RGB and perceptual loss $\mathcal{L}_Q$ evaluated through the foam renderer. This adapts the learned scene features to cell geometry, displaced surfaces, and surface-attached color. The recurrent frontend update groups and native prediction heads are optimized together. Appendix C specifies parameter initialization, stage ownership, and gradient paths.

4.2 SOURCE-CONDITIONED RADIANCE FITTING

A directional basis matched to the observations. Directional color is represented by interpolation around eight viewing axes per cell. We select these axes by farthest-point sampling from directions to source cameras and interpolated cameras. A regularized least-squares projection expresses the predicted colors in the selected basis, keeping geometry fixed. The eight-view configuration uses eight real camera centers and 21 interpolated centers in source-view order (Appendix D.1).

Real observations and independent references. A separate frozen ReSplat model reads the same sources, rendering seven reference RGB images at adjacent-pair midpoints for eight-view reconstruction. These supplement the observed photographs during fitting. All 21 interpolated centers contribute to direction selection; only the seven midpoints require RGB rendering. The reference weights remain fixed while the main frontend is trained.

Four color-fitting steps. With geometry and axes fixed, we fit colors to V real images and U reference images:

$$J(C) = \frac{\sum_{v=1}^{V} E_v^{\mathrm{real}}(C) + \lambda_{\mathrm{ref}} \sum_{u=1}^{U} E_u^{\mathrm{ref}}(C)}{V + \lambda_{\mathrm{ref}} U}, \quad E(C) = \mathrm{mean}[(\mathcal{R}_\Pi(g, a, C) - Y)^2]. \tag{4}$$

Here Y is a source photograph or predicted reference; the eight-view setting uses $V = 8$ and $U = 7$. We map image residuals to color coefficients using their rendering derivatives and normalize by accumulated observation weights to obtain an update direction D. A one-dimensional least-squares calculation selects a bounded step size γ:

$$C^+ = \begin{cases} C + \gamma D, & J(C + \gamma D) \le J(C), \\ C, & \text{otherwise.} \end{cases} \tag{5}$$

Each step accepts the proposed colors only when the rendered-image objective does not increase. Four such steps produce $\mathcal{P}_0$, the reference scene for refinement. Appendix D.1 gives the step size, normalization, and reference weight.

4.3 GEOMETRY-FIRST REFINEMENT WITH REFRESHED EVIDENCE

The refinement network (the refiner in Figure 3) uses two forms of source evidence: image residuals and gradients of the source-image loss with respect to predicted parameter updates. Residuals locate image mismatches; gradients describe how parameter changes affect those mismatches (Flynn et al., 2019; Chen et al., 2024c). We normalize these gradients and treat them as fixed input features during training. The loss and normalization are specified in Appendix C.

Let $\mathcal{D}_{\mathcal{P}_0}(s, c)$ map network outputs s and c to bounded geometry and color updates relative to $\mathcal{P}_0$. Shared spatial features f first predict s_1. The refinement network then updates geometry, recomputes source evidence, and predicts appearance:

$$\mathcal{P}_g = \mathcal{D}_{\mathcal{P}_0}(s_1, 0), \tag{6}$$

$$z_1 = \mathrm{stopgrad}\big[\mathrm{feedback}(\mathcal{P}_g, \mathcal{S}), \mathrm{adjoint}_{\mathcal{P}_0}(s_1, 0; \mathcal{S})\big], \tag{7}$$

$$c_1 = \mathrm{ColorHead}(f, z_1, \tanh s_1), \qquad \mathcal{P}_\star = \mathcal{D}_{\mathcal{P}_0}(s_1, c_1). \tag{8}$$

Here $\mathrm{feedback}$ maps image residuals to texture sites, $\mathrm{adjoint}$ computes the normalized source-loss gradients, and $\mathrm{ColorHead}$ is the appearance prediction head. Re-rendering $\mathcal{P}_g$ updates visibility and pixel-to-site contributions, so the appearance head uses evidence computed for the new geometry. The operator $\mathrm{stopgrad}$ stops gradients through these recomputed inputs during training; gradients still pass through s_1 and the final renderer. Both decodes use $\mathcal{P}_0$, applying the geometry update once. Figure 8 illustrates why source evidence changes with geometry.

4.4 TRAINING AND DEPLOYMENT

Stage I jointly trains the recurrent frontend groups and native initializer through foam target renders. Stage II trains the geometry-first refiner, initialized from the trained Stage I spatial modules, with target-view loss $\mathcal{L}_Q$. The independent RGB-reference network is fixed throughout. Color fitting uses

one ray per pixel; training evidence, source-loss gradients, and target evaluation use four subpixel rays. Test-target images are reserved for evaluation.

Construction combines learned prediction, four color-fitting steps, and geometry-first refinement. Final geometry and color are stored as one scene for rasterization and ray tracing, with backend-specific acceleration structures. New cameras render directly from this stored scene; Appendix G reports one-time construction cost and storage.

5 EXPERIMENTS

We evaluate reconstruction quality, two-view transfer to a new dataset, the construction mechanisms, and the native rendering interfaces of the recovered fields.

Data and evaluation. DL3DV140 (Ling et al., 2024; Xu et al., 2026) evaluates eight source and eight held-out target images per scene at 512×960, using the released camera selections. Scores are averaged over targets and then scenes. Training uses 9,829 DL3DV scenes, with 32 disjoint development scenes and 32 independent confirmation scenes. Two-view RealEstate10K (Zhou et al., 2018) and RealEstate10K-to-ACID (Liu et al., 2021) transfer follow ReSplat's 256×256 protocols (Xu et al., 2026). We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018), using its evaluation code. Higher PSNR/SSIM and lower LPIPS indicate better quality. Appendix E gives further details.

5.1 HIGH-QUALITY SPARSE-VIEW RECONSTRUCTION

Table 1 compares reconstruction quality under the same sparse-input protocol. SceneFoam reaches 27.97 dB PSNR, 0.873 SSIM, and 0.153 LPIPS, improving all three metrics over ReSplat K4. Its native foam also supports rasterization and ray tracing. Figure 4 shows matched held-out views and local detail.

Table 1: **Eight-view reconstruction on DL3DV140.** Published baselines (†) follow the same protocol (Xu et al., 2026). K4 denotes four recurrent updates; 4K denotes 4,000 optimization steps. Native PF acquisition comparisons and construction costs appear in Appendices E.2 and G.

Method	Representation	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat†	Gaussian	22.49	0.764	0.261
DepthSplat†	Gaussian	24.17	0.815	0.208
3DGS, 4K†	Gaussian	23.46	0.770	0.224
ReSplat, K4†	Gaussian	27.70	0.868	0.160
SceneFoam	**Foam**	**27.97**	**0.873**	**0.153**

Two-view reconstruction and cross-dataset generalization. Table 2 evaluates two-view reconstruction on RealEstate10K and transfer of the same model to ACID without ACID training or fine-tuning. SceneFoam achieves 30.08 dB, 0.916 SSIM, and 0.096 LPIPS on RealEstate10K, and 30.17 dB, 0.870 SSIM, and 0.122 LPIPS on ACID. It improves all three metrics over ReSplat in both settings. Both use three interpolated direction candidates and one midpoint RGB reference. These results demonstrate high-quality two-view reconstruction and transfer of the learned prior and source-conditioned construction to a new domain.

5.2 ACQUIRING A NATIVE FOAM: OPTIMIZATION VERSUS LEARNING

Table 11 and Figure 7 compare three native PF acquisition routes on four scenes. Both PF baselines run 30,000 updates: one uses original initialization, the other learned initialization plus virtual references. SceneFoam improves all three metrics in every scene, reaching 26.14 dB mean PSNR versus 16.65/25.22 dB. Its complete inference takes 8.94–9.44 s, compared with 36.50–53.08 minutes for PF optimization. Appendix E.2 details the quality–time comparison and the distinction between reference-supervised fitting and learned native construction.



Figure 4: **Held-out reconstruction quality, from full views to local detail.** ReSplat K4, SceneFoam, and target RGB use identical cameras and crops. The selected regions expose floating artifacts, local alignment, and thin boundaries. All target RGB is excluded from construction.

Table 2: **Two-view reconstruction and cross-dataset generalization at** 256×256**.** Left: RealEstate10K in-domain evaluation. Right: RealEstate10K-trained models evaluated on ACID without dataset-specific fine-tuning. Published baseline values (†) are reproduced from ReSplat v4, Tables 4 and 5 (Xu et al., 2026).

(a) RealEstate10K

Method	PSNR ↑	SSIM ↑	LPIPS ↓
pixelSplat†	25.89	0.858	0.142
MVSplat†	26.39	0.869	0.128
DepthSplat†	27.47	0.889	0.114
GS-LRM†	28.10	0.892	0.114
ReSplat†	29.75	0.912	0.100
SceneFoam	**30.08**	**0.916**	**0.096**

(b) RealEstate10K → ACID

Method	PSNR ↑	SSIM ↑	LPIPS ↓
pixelSplat†	27.64	0.830	0.160
MVSplat†	28.15	0.841	0.147
DepthSplat†	28.37	0.847	0.141
GS-LRM†	28.84	0.849	0.146
ReSplat†	29.87	0.864	0.135
SceneFoam	**30.17**	**0.870**	**0.122**

5.3 WHY GEOMETRY-CONDITIONED CONSTRUCTION MATTERS

Color fitting before refinement. Table 3A compares zero and four color-fitting steps using the same predictor weights and the same source-conditioned directional basis. Images are evaluated before the refinement network. This isolates the benefit of fitting appearance to the current scene's observations.

Geometry updates and evidence refresh. Table 3B compares three refinement strategies from the same fitted scene $\mathcal{P}_0$. Geometry-first evidence refresh reaches 27.97 dB, versus 27.51 dB for simultaneous updates and 27.57 dB for sequential updates retaining the original evidence. Its appearance head receives residuals and gradients recomputed for the new surfaces. This result supports updating texture evidence together with geometry, beyond changing the prediction order alone. The final row is the complete model from Table 1; Appendix F.2 details the comparison.

Table 3: **Color-fitting and refinement ablations on DL3DV140.** A: zero or four fitting steps, evaluated before refinement with identical predictor weights and directional basis. B: update order and evidence refresh, starting from the same fitted scene. The final row is the complete SceneFoam model.

A. Color fitting before refinement Color-fitting steps		PSNR ↑	SSIM ↑	LPIPS ↓
0		26.48	0.817	0.242
4		27.61	0.831	0.211
B. Refinement: update order and evidence refresh				
Update order	New evidence	PSNR ↑	SSIM ↑	LPIPS ↓
Simultaneous geometry / appearance	No	27.51	0.849	0.204
Geometry → appearance	No	27.57	0.853	0.202
Geometry → appearance	Yes	**27.97**	**0.873**	**0.153**

Conditioning the finite-step radiance fit. Table 4 compares two directional bases before and after four fitting steps using the main-model weights. Source-conditioned axes reach 27.61 dB after fitting, improving PSNR by 1.09 dB over the initial axes. Matching the basis to the observations makes the finite-step fit more effective.

Table 4: **Directional basis and color fitting.** PSNR on all 140 DL3DV140 scenes (1,120 targets), with fixed geometry and main-model weights. The initial basis uses approximately uniform Fibonacci directions; the source-conditioned basis uses camera directions.

Directional basis	No coefficient fit	Four fitting steps
Initial Fibonacci	25.92	26.52
Source-conditioned	26.48	**27.61**

Independent image references. Using the main model on DL3DV140, independent reference images raise final PSNR from 27.77 to 27.97 dB and lower LPIPS from 0.178 to 0.153. A separate reference model with fixed pretrained weights supplies these images, separating reference generation from native scene prediction. Appendix F.1 compares reference sources at the same fitting budget.

5.4 A UNIFIED RECONSTRUCTION FOR RASTERIZATION AND RAY QUERIES

The same reconstructed foam supports rasterization and ray tracing, including secondary rays reflected or refracted by inserted objects (Figure 5). Both renderers use identical scene parameters. Across all 140 DL3DV140 scenes with one view per scene and one ray per pixel, their mean absolute RGB difference is 1.479×10^{-4}. Mirrors and glass use user-specified optical properties while the reconstructed scene stays fixed.

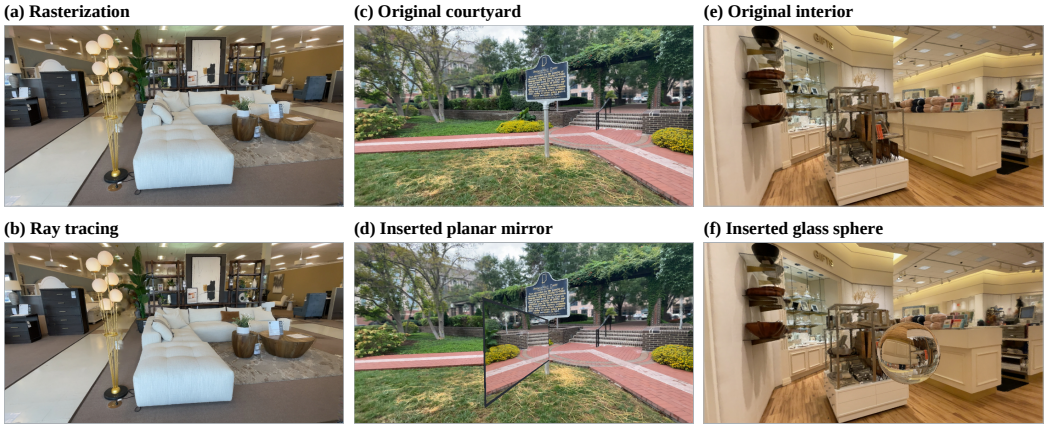

Figure 5: **Native queries on the same reconstructed fields.** Each column is a matched pair: (a,b) rasterization and ray tracing with identical one-sample pinhole rays; (c,d) a courtyard before and after mirror insertion; (e,f) an interior before and after glass insertion. The foam and camera are fixed within each pair. Mirror reflectance is 0.96; glass uses index 1.5 and two interfaces with Schlick Fresnel weights. Optical properties are user-specified.

Native non-pinhole rendering. Changing the mapping from each pixel to its viewing ray renders fisheye and radially distorted cameras from the same scene. Table 5 verifies unchanged field parameters across camera models on three reconstructed scenes. Appendix H.3 gives the camera settings and ray-validity measurements. We also simulated pull-and-release interaction with XPBD and FEM on each scene's native tetrahedral dual, directly updating cell centers and transporting surface and texture parameters (Figure 1; Appendix H.5).

Table 5: **Native camera queries from a stored foam.** Three scenes, 960×512, one ray per pixel. All camera models preserve the serialized field parameters. Valid rays satisfy the numerical ray-map criteria specified in Appendix H.3.1.

Query	Valid rays (%)	Unchanged fields
Pinhole	100.0	3/3
Equidistant fisheye, 100°	100.0	3/3
Equidistant fisheye, 150°	100.0	3/3
Radial distortion, mild	100.0	3/3

6 CONCLUSION

SceneFoam learns high-quality native Power Foam reconstruction from sparse images. Joint prediction acquires cell geometry and texture; source-conditioned fitting estimates directional radiance; and geometry-first refinement refreshes texture-site evidence before the appearance update. This geometry-consistent acquisition improves novel-view quality over ReSplat and outperforms both native-PF optimization baselines on all four comparison scenes with complete inference in about nine seconds. The recovered field is reused directly for rasterization, ray tracing, distorted cameras, and interaction on its tetrahedral dual.

REPRODUCIBILITY STATEMENT

Appendices A–D specify the construction algorithm, network coordinates, image losses and weights, public checkpoint provenance, module initialization, complete training stages, and differentiation paths. Appendices E–G specify benchmark and transfer protocols, native PF quality and timing, reference-source and refinement comparisons, and construction costs and storage. Appendix H records native-camera queries, optical settings, and deformation on the native tetrahedral dual. Source and target roles and ablation settings are specified with their protocols.

AI USE STATEMENT

Generative AI assisted manuscript organization and language editing. Experimental measurements and reconstruction images come from the authors' experiments and renderer outputs. The authors take responsibility for the manuscript, implementation descriptions, references, and results.

REFERENCES

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, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D gaussian splatting from sparse multi-view images. In *ECCV*, 2024a.

Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. MVSplat360: Feed-forward 360 scene synthesis from sparse views. In *NeurIPS*, 2024b.

Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3R: Gradient guided generalizable reconstruction. In *ECCV*, 2024c.

Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. SplatFormer: Point transformer for robust 3D gaussian splatting. In *ICLR*, 2025a.

Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Fuxin Li, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *ICCV*, 2025b.

Francesco Di Sario, Daniel Rebain, Dor Verbin, Marco Grangetto, and Andrea Tagliasacchi. Spherical voronoi: Directional appearance as a differentiable partition of the sphere. In *CVPR*, 2026.

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *CVPR*, 2019.

Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. In *ICCV*, 2025.

Shrisudhan Govindarajan, Daniel Rebain, Dor Verbin, Kwang Moo Yi, Anish Prabhu, and Andrea Tagliasacchi. Power foam: Unifying real-time differentiable ray tracing and rasterization. *arXiv preprint*, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D gaussian splatting for geometrically accurate radiance fields. *ACM Transactions on Graphics*, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *CVPR*, 2024.

Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021.

Changhe Liu, Ehsan Javanmardi, Naren Bao, Alex Orsholits, and Manabu Tsukada. UTrice: Unifying primitives in differentiable ray tracing and rasterization via triangles for particle-based 3D scenes. *arXiv preprint*, 2025a.

Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. MVSGaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *ECCV*, 2024.

Yueh-Cheng Liu, Lukas Höllein, Matthias Nießner, and Angela Dai. QuickSplat: Fast 3D surface reconstruction via learned gaussian initialization. In *ICCV*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. XPBD: Position-based simulation of compliant constrained dynamics. In *International Conference on Motion in Games*, 2016.

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 Transactions on Graphics*, 2024.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 2007.

Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022.

OpenCV. Camera calibration and 3D reconstruction. OpenCV 4.13.0 documentation, 2026a.

OpenCV. Fisheye camera model. OpenCV 4.13.0 documentation, 2026b.

Christophe Schlick. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum*, 1994.

Eftychios Sifakis and Jernej Barbič. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*, 2012.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3R: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint*, 2024.

Andrea Vedaldi and Karel Lenc. MatConvNet: Convolutional neural networks for MATLAB. In *ACM International Conference on Multimedia*, 2015.

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.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

Haifeng Wu, Wei Long, Shuhang Gu, Lixin Duan, and Wen Li. iSplat: Iterative learning for fine-grained gaussian splatting. In *CVPR*, 2026.

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

Jinlin Xiong, Can Li, Jiawei Shen, Zhigang Qi, Lei Sun, and Dongyang Zhao. FTSplat: Feed-forward triangle splatting network. *arXiv preprint*, 2026.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting gaussian splatting and depth. In *CVPR*, 2025.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. ReSplat: Learning recurrent gaussian splatting. In *ECCV*, 2026.

Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3D gaussian splatting. In *ECCV*, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 2018.

APPENDIX OVERVIEW

Appendix A defines the representation and construction algorithm. Appendices B–D describe the networks, training objectives, and color solver. Appendices E–F give evaluation protocols, native PF quality–time comparisons, reference-source and refinement comparisons, and additional images. Appendix G describes native-scene acquisition and reports construction time and field storage. Appendix H describes native rendering, ray-direction approximation, and direct simulation on the foam's tetrahedral dual. Appendix I gives an expanded comparison of representations, learned updates, and scene acquisition.

A NATIVE REPRESENTATION AND CONSTRUCTION ALGORITHM

This section details the Power Foam representation (Govindarajan et al., 2026) used by SceneFoam, its rendering equations, and the complete reconstruction algorithm.

A.1 SPATIAL TEXTURE, DIRECTIONAL APPEARANCE, AND COMPOSITING

The bounded-cell supports are defined by Eq. (1). Below, g contains positions, radii, occupied-side densities, local frames, texture-site coordinates, displacements, and spatial interpolation parameters; a contains directional axes and concentrations; C stores directional RGB coefficients.

Oriented surface and spatial texture. A local orthonormal frame (t_i, b_i, n_i) defines a base surface with normal n_i. The surface separates a side of density σ_i from an empty side. Each cell has T two-dimensional texture sites (u_{it}, v_{it}), embedded as $\xi_{it} = (u_{it}, v_{it}, 0)$ in the local frame, and displacement values h_{it}. For a world-space query x, let $\xi = r_i^{-1}[t_i, b_i, n_i]^\top (x - p_i)$ denote its radius-normalized local coordinates. Spatial interpolation gives

$$\beta_{it}(\xi) = \frac{\exp(-\kappa_i \|\xi - \xi_{it}\|^2)}{\max\{\sum_j \exp(-\kappa_i \|\xi - \xi_{ij}\|^2), 10^{-20}\}}, \qquad h_i(\xi) = \sum_t \beta_{it}(\xi) h_{it}. \tag{9}$$

Following the native Power Foam implementation (Govindarajan et al., 2026), we fix $\kappa_i = 10$ for every cell and use the squared-distance kernel with the denominator floor shown above. In world coordinates, its exponent is $-10\|x - x_{it}\|^2 / r_i^2$, where $x_{it} = p_i + r_i(u_{it} t_i + v_{it} b_i)$. The renderer evaluates displacement at the base-surface query, corrects the intersection, and recomputes the spatial weights there for appearance.

Directional appearance. At site t, K unit axes a_{itk}, concentrations τ_{itk}, and RGB coefficients C_{itk} define

$$\begin{aligned} \phi_{itk}(d) &= \frac{e^{-\tau_{itk}\|d - a_{itk}\|}}{\max\{\sum_j e^{-\tau_{itj}\|d - a_{itj}\|}, 10^{-8}\}}, \quad c_{it}(d) = \max\left(0, \tfrac{1}{2} + \sum_k \phi_{itk}(d) C_{itk}\right), \\ c_i(\xi; o) &= \sum_t \beta_{it}(\xi) c_{it}(d_{it}(o)). \end{aligned} \tag{10}$$

For camera origin o, $d_{it}(o) = (x_{it} - o)/\|x_{it} - o\|$, where $x_{it} = p_i + r_i(u_{it} t_i + v_{it} b_i)$ is the site's base-plane point; the maximum acts per RGB channel. Spatial weights β select surface positions; directional weights ϕ select viewing directions (Di Sario et al., 2026).

The ray-segment origin convention and a conditional error bound for site-based directions are given in Appendix H.4.

Ray compositing. For ordered occupied ray segments of lengths ℓ_i, opacity is $\alpha_i = 1 - e^{-\sigma_i \ell_i}$ and transmittance is $T_i = \exp(-\sum_{j<i} \sigma_j \ell_j)$. Pixel color is $\sum_i T_i \alpha_i c_i + T_{\text{end}} c_{\text{bg}}$. Geometry changes intersections, segment lengths, and visibility; appearance changes segment colors. Rasterization and adjacency traversal organize intersection computation differently while reading the same geometry and radiance parameters (Govindarajan et al., 2026; 2025).

A.2 NOTATION AND THE COMPLETE CONSTRUCTION ALGORITHM

Table 6: **State and parameter glossary.** $N = 245{,}760$, $T = 8$, and $K = 8$ in the eight-view configuration. The field stores continuous parameters; the renderer additionally constructs topology and acceleration data.

Symbol	Shape	Meaning
$\mathcal{S}$	V RGB/camera pairs	Real observations available during construction.
G_{adapt}, H	scene, $N \times 512$	Jointly trained frontend prediction and hidden state.
G_{ref}	Gaussian scene	Separate frozen-reference prediction.
p, r, σ	$N \times (3 + 1 + 1)$	Cell centers, radii, and occupied-side densities.
(t, b, n)	local frames	Oriented surface coordinate systems.
ξ, h	$N \times T \times (2 + 1)$	Local texture sites and normal displacements.
a, τ, C	per site and axis	Unit directions, concentrations, and RGB coefficients.
s, c	$N \times 16$, $N \times T \times 24$	Geometry and appearance decoder outputs.
$\mathcal{P}_{\mathrm{seed}}, \mathcal{P}_{\mathrm{init}}$	native fields	Seed scene and learned initial scene.
$\mathcal{P}_{\mathrm{dir}}$	native field	Source-conditioned basis before coefficient fitting.
$\mathcal{P}_0, \mathcal{P}_g, \mathcal{P}_\star$	native fields	Fitted scene, geometry-updated scene, and final scene.

Algorithm 1: constructing a field from observations.

1. Read the source photographs and calibrated cameras $\mathcal{S}$, and normalize the cameras using the frontend's coordinate convention.
2. Run the jointly trained frontend for four updates to obtain G_{adapt} and H. Independently run the fixed reference model for four updates on the same sources to obtain G_{ref}.
3. Transfer adapted means to cell centers. Construct radii, densities, frames, eight local sites, and the source-projected directional fit to obtain $\mathcal{P}_{\mathrm{seed}}$.
4. Render $\mathcal{P}_{\mathrm{seed}}$ at real source cameras, lift residuals, and evaluate the learned initializer to obtain $\mathcal{P}_{\mathrm{init}}$.
5. Form 21 virtual cameras from adjacent source pairs. Use their centers and the eight real centers to select eight directional axes. Re-express the angular function by a sitewise ridge solve.
6. Render seven midpoint RGB images from G_{ref}, one per adjacent source pair. Evaluate four color-fitting steps using the eight real and seven reference images. Apply the acceptance test in Eq. (31); the resulting field is $\mathcal{P}_0$.
7. Render real sources at $\mathcal{P}_0$, lift residuals, and compute normalized source-loss gradients in decoder-output coordinates. Evaluate the spatial blocks and geometry head to obtain f, s_1.
8. Decode $\mathcal{P}_g = \mathcal{D}_{\mathcal{P}_0}(s_1, 0)$. Re-render sources, recompute residuals and source-loss gradients, and detach the refreshed features z_1.
9. Predict $c_1 = \mathrm{ColorHead}(f, z_1, \tanh s_1)$ and decode $\mathcal{P}_\star = \mathcal{D}_{\mathcal{P}_0}(s_1, c_1)$. Serialize the field and prepare the requested renderer's acceleration structure.
10. For each new camera, render $\mathcal{P}_\star$ without source RGB or reconstruction-network evaluation.

The two Gaussian predictions serve different roles: the jointly trained branch supplies native structure and conditions the initializer; the independently pretrained branch supplies RGB references. Inference includes both network evaluations and the seven reference-image renders.

Geometric meaning of the cell boundaries. Expanding the pairwise inequality in Eq. (1) gives

$$2(p_j - p_i)^\top x \le \|p_j\|^2 - r_j^2 - \|p_i\|^2 + r_i^2. \tag{11}$$

Thus pairwise interfaces lie on planes, and the additional sphere clips each domain. Here "power" names the squared-distance criterion in Eq. (1). Cell centers define spatial support; separate surface sites define texture; directional coefficients interpolate appearance over viewing directions.

B NETWORK ARCHITECTURE AND BOUNDED PARAMETER UPDATES

This section specifies the seed scene, network inputs, bounded parameter updates, and geometry-first refinement. The lower row of Figure 3 illustrates the corresponding decoder-level flow.

B.1 NATIVE INITIALIZATION AND NETWORK DETAILS

Seed geometry and projected appearance. Gaussian means initialize PF centers. Initial radii are $r_i = 0.25 d_{i,64}$, where $d_{i,64}$ is the 64th-neighbor distance, and initial densities are $4/r_i$. The source-oriented eigenvector associated with the smallest covariance eigenvalue is replaced by a normal estimated from the frontend-rendered depth where that normal is valid. For each cell, the center texture site has coordinates $(0, 0)$; the other seven are uniformly distributed around a radius-0.5 circle. Initial height is zero. Eight approximately uniform Fibonacci directions with initial concentration two define the initial directional basis. If $h_{i,0} \in \mathbb{R}^3$ is the Gaussian's constant (degree-zero) spherical-harmonic coefficient, the initial color coefficient at every site and axis of cell i is

$$(C_a)_{itk} = \mathrm{clip}(0.28209479177387814\, h_{i,0} + \tfrac{1}{2}, 0, 1) - \tfrac{1}{2}. \tag{12}$$

Thus C_a denotes coefficients before the shader's 0.5 RGB offset.

For a site x_t projected into source view v, let α_v and D_v be opacity and depth rendered from the frontend prediction, and z_{vt} its projected depth. The initial appearance fit uses

$$w_{vt} = \mathbf{1}_{\mathrm{valid}} \alpha_v(\Pi_v x_t) \exp\left(-\frac{|z_{vt} - D_v(\Pi_v x_t)|}{\max(r_i, 10^{-6})} \right). \tag{13}$$

The validity mask requires positive depth and an in-image projection. With directional design matrix Φ, diagonal weights W, sampled source colors Y and initial coefficients C_a, the coefficients solve

$$(\Phi^\top W \Phi + \lambda I) C = \Phi^\top W (Y - \tfrac{1}{2}) + \lambda C_a, \qquad \lambda = 0.02 \max\left(\sum_v w_{vt}, 1 \right). \tag{14}$$

The initial local fit applies a coefficient lower bound of -0.5. The subsequent four-step radiance fit uses objective-based acceptance with unconstrained coefficients.

Native initializer. The frontend provides a 512-channel hidden state. An additional pass through its error-processing blocks uses source images rendered from the seed foam to compute features for the initializer. These features, rather than the associated Gaussian prediction, condition the foam updates. Its explicit cell features have 421 channels, including positional encoding, native geometry, covariance/opacity cues, appearance and source feedback. Earlier copied spatial blocks use rank-eight linear adapters; the last copied block is trainable. A shared feature network predicts 16 geometry-update and 192 appearance-update values per cell. A per-site prediction head combines a 64-channel cell embedding with 87 site features.

The 87 features comprise 24 RGB directional coefficients, 24 directional residuals, eight transformed observation weights, 24 axis coordinates and seven geometry values. An observation weight sums the rendering contributions assigned to a site. The geometry values are two local coordinates, normalized height, three normal coordinates and $\log(1 + \sigma r)$. In addition to Eq. (18), height changes are $0.5 r_i \tanh s_{it,h}$ and normal increments are $0.5 \tanh s_{i,n}$, followed by frame transport. Texture height enters the native geometry separately from the shader-site coordinates.

Transport of the texture frame. Let q_i be the current unit scalar-first quaternion, n_i^q the normal extracted from its frame, and n_i the stored normal. Define $\nu(x) = x / \max(\|x\|_2, 10^{-8})$ and $v_i = 0.5 \tanh s_{i,n}$. The implemented transport is

$$\delta_i = \nu(n_i^q + v_i) - \nu(n_i^q), \tag{15}$$

$$q_i' = q_i \otimes \nu\left(\begin{bmatrix} 1 + (n_i^q)^\top (n_i^q + \delta_i) \\ -n_i^q \times \delta_i \end{bmatrix} \right), \qquad n_i' = n_i + \delta_i. \tag{16}$$

Here $\otimes$ is quaternion multiplication. Power Foam uses rows of the quaternion rotation matrix for its frame, so the inverse shortest rotation is composed on the right. This transports the existing texture chart with the normal instead of reconstructing a canonical tangent frame. The bounded increment satisfies $\|v_i\|_2 < \sqrt{3}/2$, avoiding an antipodal normal update.

B.2 Bounded native decoder coordinates

The initializer and refinement network predict parameter increments through bounded mappings. Their main updates are

$$p_i' = p_i + 0.5 r_i \tanh s_{i,p}, \qquad r_i' = r_i \exp(0.69 \tanh s_{i,r}), \tag{17}$$

$$\sigma_i' = \sigma_i \exp(2 \tanh s_{i,\sigma}), \qquad C_{itk}' = C_{itk} + 0.5 \tanh c_{itk}. \tag{18}$$

Normal updates transport the texture frame; site heights change by at most $0.5 r_i$ in each decode. The initializer predicts a complete native field. During refinement, both geometry and appearance updates are measured relative to the fitted scene $\mathcal{P}_0$.

B.3 Geometry and appearance refinement details

Geometry-conditioned appearance prediction. Both geometry and appearance updates are decoded relative to the fitted scene $\mathcal{P}_0$. The geometry head predicts s_1, producing $\mathcal{P}_g = \mathcal{D}_{\mathcal{P}_0}(s_1, 0)$. Re-rendering this scene yields the evidence used by the appearance head:

$$z_1 = \mathrm{stopgrad}[\mathrm{feedback}(\mathcal{P}_g, \mathcal{S}), \mathrm{adjoint}_{\mathcal{P}_0}(s_1, 0; \mathcal{S})], \tag{19}$$

$$c_1 = \mathrm{ColorHead}(f, z_1, \tanh s_1), \qquad \mathcal{P}_\star = \mathcal{D}_{\mathcal{P}_0}(s_1, c_1). \tag{20}$$

Directional re-expression and four color-fitting steps precede refinement. At their output, the update variables are initialized to $s_0 = c_0 = 0$. The initial source residuals and gradients from Eq. (24) pass through input-encoding, error-processing, and spatial feature blocks to produce f and then s_1. After the geometry update, the renderer rebuilds its adjacency graph and recomputes residuals and gradients at $(s_*, c_*) = (\mathrm{stopgrad}(s_1), 0)$, still relative to $\mathcal{P}_0$.

The appearance head reuses f and receives these recomputed measurements together with $\tanh s_1$. During training, the measurements are treated as constants, while s_1 remains differentiable. The final scene has the geometry of $\mathcal{P}_g$ and color coefficients $C_0 + 0.5 \tanh c_1$. Thus geometry is updated once, and appearance is predicted from evidence for that updated geometry.

C Training objectives, configuration, and differentiation

C.1 Training losses and color-fitting objective

The training configuration uses an RGB reconstruction term and a perceptual feature term:

$$\mathcal{L}_{\mathrm{img}}(\hat{I}, I) = \lambda_{\mathrm{rgb}} \mathcal{L}_{\ell_1}(\hat{I}, I) + \lambda_{\mathrm{VGG}} \mathcal{L}_{\mathrm{VGG}}(\hat{I}, I), \qquad (\lambda_{\mathrm{rgb}}, \lambda_{\mathrm{VGG}}) = (1, 0.5). \tag{21}$$

The target loss $\mathcal{L}_Q$ averages this supervision over training target views; the source-gradient objective averages it over real source views. The color solver uses the RGB mean-squared error J in Eq. (4); its acceptance test enforces a non-increasing fitting objective.

Table 7: **Training losses and coefficients.** Image supervision and coefficient fitting have distinct data roles and objectives. LPIPS is the perceptual evaluation metric.

Objective or coefficient	Definition / value
Stage I direct target supervision	$\mathcal{L}_{\ell_1} + 0.5\mathcal{L}_{\mathrm{VGG}}$
Stage I optional auxiliary term	$\lambda_{\mathrm{aux}} = 0.5$ during Stage I training; fitted axes and colors fixed in backpropagation
Stage II target supervision	$\mathcal{L}_{\ell_1} + 0.5\mathcal{L}_{\mathrm{VGG}}$
Real-source gradient objective	$\mathcal{L}_{\ell_1} + 0.5\mathcal{L}_{\mathrm{VGG}}$; real sources only
Fixed-geometry coefficient objective	Per-view RGB MSE; real-view weight 1; $\lambda_{\mathrm{ref}} = 1.2$
Perceptual feature implementation	MatConvNet VGG19 inherited from ReSplat

Optional geometry regularization. Stage I training includes an optional auxiliary target-view term evaluated after color fitting:

$$\mathcal{L}_{\text{pred}} = \mathcal{L}_Q(g_\psi, a_\psi, C_\psi) + \lambda_{\text{aux}} \mathcal{L}_Q\big(g_\psi, \text{stopgrad}(a_{\text{fit}}), \text{stopgrad}(C_{\text{fit}})\big). \tag{22}$$

Here ψ denotes the trainable frontend and initializer weights. The first term supervises the predicted scene; the auxiliary term supervises geometry while holding fitted directions and colors fixed during backpropagation. The solver and reference model remain outside this gradient path. This training-only regularizer can be disabled with $\lambda_{\text{aux}} = 0$ without changing scene construction. Table 7 gives the training coefficient.

C.2 Trainable components and construction inputs

Stage I jointly updates the five recurrent frontend groups and the native initializer. Stage II trains a separate geometry-first refiner using the locked Stage I predictor. The initial image encoder and independent RGB-reference network retain their pretrained weights; the reconstruction frontend update groups are learned jointly with the initializer in Stage I. During scene construction, both frontend paths run on the same real sources; the learned parameters are fixed while native scene parameters are predicted, fitted, and refined. After export, the renderer uses the stored foam and its acceleration data.

Field-parameter ownership. The state variables g, a, C describe the complete native field, not a list of independently learned outputs; Table 8 specifies their updates during reconstruction. The local texture coordinates $\xi_{it} = (u_{it}, v_{it}, 0)$ retain the canonical layout in Appendix B.1, while their world-space positions $x_{it} = p_i + r_i(u_{it}t_i + v_{it}b_i)$ move with the predicted cell center, radius, and frame. Directional axes and concentrations are computed from source-camera coverage rather than learned by gradient descent, with $a_{itk} = a_{ik}$ and $\tau_{itk} = \tau_i$ shared across sites within a cell, whereas C_{itk} remains distinct for each site, axis, and RGB channel.

Table 8: **Updates of native field parameters during reconstruction.** Predict denotes an output of a learned network with fixed weights at construction time; fixed denotes no update in that operation. FPS and the concentration rule use source-camera geometry (Appendix D); projection and fitting are numerical coefficient updates. The table concerns reconstruction, separately from deformation transport in Appendix H.5.

Field parameter	Initializer	Direction basis	Color fitting	Refiner	Sharing
Center, radius, density p_i, r_i, σ_i	Predict	Fixed	Fixed	Predict	Per cell
Normal / local frame	Predict	Fixed	Fixed	Predict	Per cell
Heights h_{it}	Predict	Fixed	Fixed	Predict	Per site
Local sites ξ_{it}	Fixed	Fixed	Fixed	Fixed	Canonical layout
Spatial kernel $\kappa_i = 10$	Fixed	Fixed	Fixed	Fixed	Global
Axes a_{ik}	Fixed	FPS	Fixed	Fixed	Per cell
Concentration τ_i	Fixed	Coverage rule	Fixed	Fixed	Per cell
RGB coefficients C_{itk}	Predict	Ridge projection	Fit	Predict	Per site / axis

C.3 Model initialization, training sequence, and differentiation paths

Public checkpoint and branch ownership. We use the official ReSplat DL3DV 8-view checkpoint (Xu et al., 2026), with SHA-256 prefix `8179ed87`.[1] Separate parameter copies initialize the reconstruction and RGB-reference branches. During Stage I, the reconstruction branch's five recurrent update groups are trained jointly with the native initializer through the PF rendering loss. The initial image encoder retains its pretrained weights, and the independent reference branch preserves its own pretrained parameters throughout.

[1] Checkpoint `resplat-base-dl3dv-512x960-view8-8179ed87.pth` in the official model listing.

Stage I: joint native prediction. The initializer's spatial-processing blocks are independent copies of the point-space update modules in the public ReSplat checkpoint. Its new PF geometry, appearance, and texture readouts use Kaiming initialization for hidden layers (He et al., 2015) and zero-initialized final residual outputs. Each new rank-eight LoRA adapter (Hu et al., 2022) has one randomly initialized factor and one zero factor, so its initial weight increment is zero. These initializations precede the four Stage I epochs; their residual heads and adapters are then learned together with the designated frontend groups.

Stage II: geometry-first refinement. After the four Stage I epochs, we lock that run's trained predictor and copy its spatial-processing modules, including the corresponding learned adapters, into the separate geometry-first refiner. These copies are trainable and have no shared parameters with the locked predictor. The new error encoders and geometry/appearance readouts are initialized afresh, with zero-initialized final feedback-injection and residual-output layers. Stage II trains this refiner for eight epochs. The native-model lineage is the public ReSplat checkpoint, this run's four-epoch Stage I predictor, and its eight-epoch Stage II refiner; no earlier native-model checkpoints are loaded.

Training data and target coverage. We trained both stages on the same fixed set of 9,829 valid DL3DV training scenes. We reserved 32 disjoint development scenes for model selection and another 32 scenes as an independent confirmation set. Each training sample contained eight ordered, calibrated source images at 512×960 and two targets drawn from a fixed pool of eight target views for that scene. This initialization and the two-stage schedule specify the complete training of the DL3DV native initializer and refiner from the public ReSplat starting point. Model selection used only the development set; the selected weights and construction configuration were then held fixed for the official 140-scene evaluation.

Stage I jointly trained the native initializer and five recurrent frontend update groups for four epochs, covering each scene's eight targets once. Stage II trained the geometry-first refiner for eight epochs using the locked Stage I predictor, covering them twice (Table 9). Both stages used two H800 GPUs; the initial image encoder and independent RGB-reference network remained frozen. Both stages used the losses in Table 7, including ReSplat's MatConvNet VGG19 implementation (Simonyan & Zisserman, 2015; Vedaldi & Lenc, 2015).

Gradient paths. Target-image losses differentiate continuous scene parameters through the Power Foam renderer. Adjacency-graph construction is outside backpropagation, and recurrent states use the frontend's existing stop-gradient operations. When backpropagating through residual mapping in the initializer, rendering weights, accumulated observation weights, and directional pooling weights are held fixed; gradients pass through the negative rendered image in the residual. The per-scene color solver lies outside the differentiated path. The initial-field loss supervises geometry and appearance jointly; the optional auxiliary term follows the stopped-gradient path defined in Eq. (22).

Source-loss gradient coordinates and normalization. Let $\mathcal{D}_{\mathcal{P}_0}(s, c)$ denote the anchored field decoder, with $s \in \mathbb{R}^{N\times 16}$ and $c \in \mathbb{R}^{N\times 8\times 24}$. Its source objective uses only the V real views ($V = 8$ for DL3DV and $V = 2$ for RealEstate10K) and the four-ray renderer $\mathcal{R}^{(4)}$:

$$E_{\mathcal{S}}(s, c) = \frac{1}{V}\sum_{v=1}^{V} \mathcal{L}\Big(\mathcal{R}^{(4)}(\mathcal{D}_{\mathcal{P}_0}(s, c); \Pi_v), I_v\Big), \tag{23}$$

$$g^s = \left.\frac{\partial E_{\mathcal{S}}}{\partial s}\right|_{(s_*, c_*)}, \qquad g^c = \left.\frac{\partial E_{\mathcal{S}}}{\partial c}\right|_{(s_*, c_*)}. \tag{24}$$

Here $\mathcal{L}$ is the same RGB $\ell_1 + 0.5\mathcal{L}_{\mathrm{VGG}}$ loss used for learning. Temporary copies of s_*, c_* are treated as independent differentiable variables. Gradients from the real source renders are accumulated with respect to these variables. Thus all gradient features describe parameter updates relative to the same scene $\mathcal{P}_0$.

For geometry coordinate j and appearance coordinate a, the normalization scales are

$$r^s_j = \max\left(\sqrt{\frac{1}{N}\sum_i (g^s_{ij})^2}, 10^{-12}\right), \qquad r^c_a = \max\left(\sqrt{\frac{1}{8N}\sum_{i,t} (g^c_{ita})^2}, 10^{-12}\right). \tag{25}$$

Table 9: **Complete two-stage DL3DV training.** Both stages used the same fixed training and development splits. Each stage epoch visits every training scene once, with two targets per visit; scene-sample counts exclude sampler padding.

Setting	Stage I: Joint native prediction	Stage II: Geometry-first refinement
Training scenes	9,829	9,829
Input resolution	512×960	512×960
Source / target views per sample	8 / 2	8 / 2
Stage-specific epochs	4	8
Optimizer updates per epoch	1,229	1,229
Total optimizer updates	4,916	9,832
Scene samples in stage	39,316	78,632
Target-image supervision count	78,632	157,264
Complete target-pool passes	1	2
GPUs	2× H800	2× H800
Microbatch per GPU	1 scene	2 scenes
Gradient-accumulation steps	4	2
Nominal effective batch	8 scenes / 16 targets	8 scenes / 16 targets
Trainable modules	Native initializer; five frontend update groups	Geometry-first refiner
Unchanged in this stage	Initial image encoder; independent RGB reference	Locked Stage I predictor; independent RGB reference
Trainable parameters	23,404,397	9,768,872
Optimizer	AdamW	AdamW
Adam $(\beta_1, \beta_2, \epsilon)$	$(0.9, 0.999, 10^{-8})$	$(0.9, 0.999, 10^{-8})$
Weight decay	10^{-4}	10^{-4}
Global gradient-norm limit	1	1

Each corresponding element is transformed by

$$\widetilde{g} = \mathrm{stopgrad}[\mathrm{clip}(\mathrm{sign}(g)\log(1 + |g|/r), -8, 8)]\,. \tag{26}$$

We write $\mathrm{adjoint}_{\mathcal{P}_0}(s_*, c_*; \mathcal{S}) = (\widetilde{g}^s, \widetilde{g}^c)$ for this normalized, detached feature pair. Geometry gradients are normalized by their root-mean-square (RMS) magnitude over cells. Color gradients use an RMS over cells and sites, separately for each of the 24 coordinates. These normalized features are stored in half precision. The refiner treats these detached first-order gradients as input features. In the geometry-first variant, differentiable geometry-update variables also enter the appearance head, so final target loss can affect geometry through both the render and the appearance-prediction path. Topology construction and hard field-of-view or visibility selection remain outside backpropagation.

Optimizer and schedule. Both stages used AdamW (Loshchilov & Hutter, 2019), with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 10^{-4}, and global gradient-norm clipping at one. Stage I learning rates were 9×10^{-6} for spatial blocks, 9×10^{-5} for the shared feature network and appearance/site heads, 3×10^{-5} for geometry, and 3×10^{-6} for each frontend update group. Stage II used 3×10^{-5} for prediction heads, error-processing blocks, and input encoders, and a final spatial rate of 9×10^{-6}. Each stage created a fresh AdamW optimizer, including fresh moment estimates and update counters. A within-stage resume restored that stage's optimizer state, random-number-generator state, and data-sampling position. Optimizer states remain stage-specific. At effective batch size eight, each stage epoch comprised $\lceil 9{,}829/8 \rceil = 1{,}229$ updates. The prediction and refinement stages used 4,916 and 9,832 updates, totaling 14,748 over $4 + 8 = 12$ stage-specific epochs. Scene-sample counts exclude sampler padding. Construction used four color-fitting steps, independently pretrained RGB references, and the source-conditioned direction concentration rule.

Development metrics. PSNR, SSIM and LPIPS use the released ReSplat metric implementation. Each metric compares rendered RGB to the target at 512×960. Development evaluation averages over eight targets per scene and then equally over the 32 scenes.

D DIRECTIONAL CONDITIONING AND NUMERICAL RADIANCE FITTING

This section gives the directional-basis projection, numerical color updates, and a fitting-step comparison with fixed network weights. The solver minimizes RGB mean-squared error.

D.1 DIRECTIONAL RE-EXPRESSION AND SOURCE-CONDITIONED RADIANCE FITTING

Directional re-expression. Eight real camera centers and 21 source-derived virtual camera centers form the direction candidates. Source cameras follow the input trajectory order. For each adjacent pair $v, v+1$, $v = 1, \dots, 7$, three cameras use fractions $\eta \in \{1/4, 1/2, 3/4\}$. For camera-to-world rotation R_v, translation t_v, and intrinsics K_v, spherical linear interpolation (Slerp) gives

$$\begin{aligned} R_{v,\eta} &= \mathrm{Slerp}(R_v, R_{v+1}; \eta), \\ t_{v,\eta} &= (1-\eta)t_v + \eta t_{v+1}, \qquad K_{v,\eta} = (1-\eta)K_v + \eta K_{v+1}. \end{aligned} \tag{27}$$

The reference model renders only the seven midpoint cameras ($\eta = 1/2$). Color fitting uses all seven resulting RGB images; the remaining interpolated cameras supply direction candidates without RGB rendering. Virtual RGB targets come from the independent ReSplat reference model after four recurrent updates, whose weights stay frozen independently of the adapted PF frontend. This reference branch receives the same eight source images and cameras. It renders Gaussian RGB directly, without a native initializer or refiner. The PF field, its directional basis and spatial features come from the adapted reconstruction branch. The color objective weights each real view by 1 and each virtual view by 1.2.

At each cell center, deterministic farthest-point sampling begins at the first source direction and selects eight axes. If ρ_i is the largest distance to the selected set among the 29 unit directions, their shared concentration is $\tau_i = \mathrm{clip}(2/\max(\rho_i, 10^{-3}), 2, 32)$. The stored raw axis is $\tau_i a_{ik}$; the shader recovers concentration as its norm (floored at 10^{-6}) and divides the raw axis by that concentration. Axes are chosen at cell centers; fitting below uses directions evaluated at the actual sites. The coverage statistic is evaluated at cell centers.

For each site, let $\Phi \in \mathbb{R}^{29\times 8}$ contain the new basis weights and let $Y \in \mathbb{R}^{29\times 3}$ contain the old coefficient sum evaluated at those directions, before adding the 0.5 offset and lower RGB clamp. Let $\bar{C}$ repeat the mean old coefficient across the eight axes. The new coefficients satisfy

$$C^{\mathrm{dir}} = (\Phi^\top \Phi + \lambda I)^{-1}(\Phi^\top Y + \lambda \bar{C}), \qquad \lambda = 10^{-4} \max\left(\frac{\mathrm{tr}(\Phi^\top \Phi)}{8}, 10^{-6}\right). \tag{28}$$

The inverse denotes a small linear solve. This approximates the existing directional function in a source-conditioned basis. Geometry is fixed during this approximate angular projection.

Residual lifting and gain. The color solver uses the eight source photographs and seven midpoint reference images. Let $e_v = Y_v - \mathcal{R}_v(C)$ and let the native lift provide residual mean ℓ_{vt} and contribution mass m_{vt} at site t. For directional basis weights b_{vtk} masked by the camera field of view and RGB channel c, the lower-clamp derivative is $a_{vtc} = \mathbf{1}[0.5 + \sum_k b_{vtk} C_{tkc} > 0]$. The preconditioned direction is

$$D_{tkc} = \frac{\sum_v w_v m_{vt} b_{vtk} a_{vtc} \ell_{vtc}}{\max\{\sum_v w_v m_{vt} b_{vtk}, 10^{-8}\}}. \tag{29}$$

The denominator is a positive row-sum preconditioner. Native rasterization of site perturbations $\sum_k b_{vtk} a_{vtc} D_{tkc}$ produces the image perturbation q_v. The implementation computes

$$\gamma = \mathrm{clip}\left(\frac{\sum_v w_v \langle q_v, e_v\rangle}{\max\{\sum_v w_v \|q_v\|^2, 10^{-20}\}}, 0, 4\right), \tag{30}$$

$$C^+ = \begin{cases} C + \gamma D, & J(C+\gamma D) \le J(C), \\ C, & \text{otherwise.} \end{cases} \tag{31}$$

Both objective values use the actual clamped renderer. If evaluation is finite and deterministic, Eq. (31) cannot increase the evaluated scalar J: acceptance requires a non-increase, and rejection leaves C unchanged. The property concerns the source objective in Eq. (4). Radiance fitting uses its configured one-ray render/lift path; training, source-loss gradients and reported target scores use four subpixel rays. The shader gives out-of-frustum sites a constant 0.5 fallback and zero coefficient derivative.

D.2 RADIANCE-FITTING ANALYSIS

The main-model sweep uses all 140 DL3DV140 scenes (1,120 targets) at 0, 1, 2, 4 and 8 coefficient proposals, with network weights fixed. Figure 6 plots the real/reference objective in Eq. (4) and held-out quality, with the same refiner applied at 2, 4 and 8 proposals.

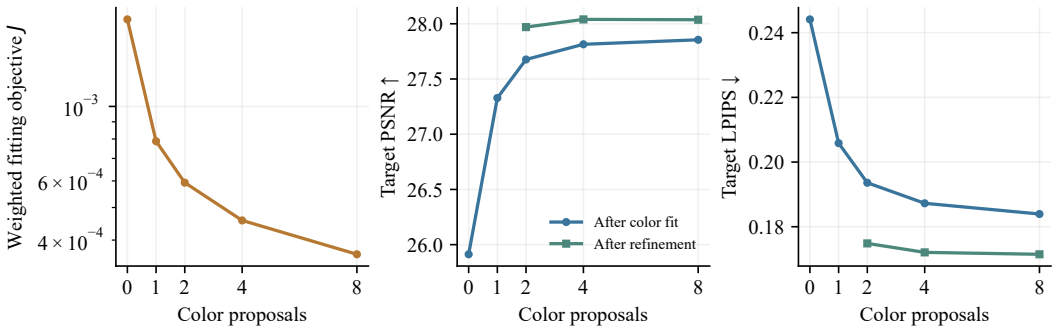

Figure 6: **Fitting and held-out quality on DL3DV140.** All 140 scenes and 1,120 targets; main-model weights as in Tables 4 and 12. The objective uses real/reference images; target metrics use held-out images. "After refinement" applies the same refiner.

E EVALUATION PROTOCOLS AND NATIVE ACQUISITION COMPARISONS

E.1 DATASETS AND EVALUATION SETTINGS

Eight-view DL3DV140 evaluation. Eight-view reconstruction and mechanism studies use the full released DL3DV140 manifest: 140 scenes, eight held-out targets per scene (1,120 in total), and 512×960 resolution. Each image is scored using the released ReSplat metric implementation, then averaged within scenes and equally over all 140 scenes. The complete model obtains 27.96661 dB PSNR, 0.873011 SSIM, and 0.152866 LPIPS; tables use two decimal places for PSNR and three for SSIM and LPIPS. All component rows use the same held-out cameras. Target RGB is excluded from construction.

Development set and controlled studies. Model selection used 32 development scenes with eight targets each; another 32 scenes formed an independent confirmation set. Tables 4 and 12 and Figure 6 use the main-model weights from Tables 1 and 3 on the full DL3DV140 set. Inference, storage, and large-scale backend agreement also cover all 140 scenes. The camera sweep uses three stored fields. Weights and cameras stay fixed within each comparison.

Two-view RealEstate10K. Table 2a follows the released two-view 256×256 protocol. After checking data validity, we fixed a split of the 66,033-scene training index into 65,521 training scenes and 512 development scenes. Each scene sample contained two ordered, calibrated source images and four target images distinct from both sources. Both adaptation stages ran for four complete epochs, with the trainable modules, frozen modules, image losses, optimizer, and group-specific learning rates specified in Appendix C. Stage I jointly trained the native initializer and selected frontend update groups; Stage II trained the refiner using the locked Stage I predictor. Table 10 gives the batch and update counts.

We evaluated development performance every 1,000 optimizer updates and at each epoch end. Model selection used only the RealEstate10K development set. Each stage comprised $4\lceil 65{,}521/8 \rceil = 32{,}764$ optimizer updates. Across both stages, adaptation used 65,528 updates and 524,168 scene samples, excluding sampler padding. At construction time, three interpolated cameras supplied direction candidates, and only the midpoint camera was rendered to supply one reference RGB image; the initial Fibonacci basis supplied additional axes when needed. Real-image and reference-image weights were 1 and 1.2, respectively.

Table 10: **Two-stage RealEstate10K adaptation.** Each stage used all 65,521 training scenes for four complete epochs. Scene-sample counts exclude sampler padding; each scene sample supervised four target images. Model selection used only the 512-scene development set.

Setting	Stage I: native prediction	Stage II: refinement
Input resolution	256×256	256×256
Source / target views per sample	2 / 4	2 / 4
Full epochs	4	4
Optimizer updates per epoch	8,191	8,191
Total optimizer updates	32,764	32,764
Scene samples over four epochs	262,084	262,084
Target-image supervision count	1,048,336	1,048,336
GPUs	2× H800	2× H800
Microbatch per GPU	1 scene	1 scene
Gradient-accumulation steps	4	4
Nominal effective batch	8 scenes / 32 targets	8 scenes / 32 targets

RealEstate10K-to-ACID generalization. Table 2b follows ReSplat v4's two-view, 256×256 cross-dataset protocol (Xu et al., 2026). The RealEstate10K-trained SceneFoam model, including its independent reference branch, is applied to ACID without ACID training, fine-tuning, or checkpoint selection. Construction still uses the two permitted ACID source images for prediction, reference generation, and coefficient fitting; held-out ACID RGB is used only for scoring. This distinguishes scene-conditioned coefficient fitting from updating network weights on the target dataset.

Sources of baseline results. The Gaussian baseline scores for DL3DV140 in Table 1 are from ReSplat v4, Table 1; the RealEstate10K and ACID baselines in Table 2 are from its Tables 4 and 5, respectively (Xu et al., 2026). SceneFoam is evaluated under the same protocol as the baselines in each table.

E.2 Native Power Foam acquisition: quality and time

Table 11 and Figure 7 compare three acquisition routes across the visitor information room, bookshelf interior, pavilion and skyline, and garden and timber shed. Within each scene, all methods use the same sparse observations and held-out evaluation camera. Both native PF baselines follow the per-scene optimization procedure of Power Foam (Govindarajan et al., 2026) for 30,000 updates. The first uses original initialization; the second uses learned initialization and augments supervision with virtual RGB references. SceneFoam uses its fixed pretrained model and prediction–fitting–refinement procedure.

Sparse-view reconstruction quality. Original-initialization PF reaches a mean PSNR of 16.6549 dB after 30,000 updates. Learned initialization with virtual references raises it to 25.2242 dB. SceneFoam reaches 26.1401 dB, 0.8406 SSIM, and 0.2000 LPIPS, improving all three metrics in every scene over both baselines. Relative to reference-assisted PF, its mean gains are 0.9159 dB PSNR and 0.0374 SSIM, with a 0.0377 reduction in LPIPS. Learning the native reconstruction thus improves sparse-view quality beyond the gains from initialization and additional image supervision.

Native acquisition time. Original-initialization PF takes 38.20–53.08 minutes of optimization; learned initialization with virtual references takes 36.50–44.03 minutes. Complete SceneFoam inference takes 8.94–9.44 s. The reference-assisted PF optimization averages 2,363.685 s, compared with 9.235 s for complete SceneFoam inference, a 256-fold ratio of the reported mean times. Virtual references improve the PF fit while retaining its per-scene reconstruction cost.

From Gaussian cues to a native field. Gaussian centers and depth provide cues for initial cell positions and surface orientations (Appendix B.1). A native PF field additionally specifies bounded cell supports, internal surface displacements, local texture sites, and per-site directional radiance. Using rendered Gaussian views to supervise these variables is a new image-based PF reconstruction,

Table 11: **Native PF acquisition on four scenes.** PF opt. reports the 30,000-step optimization time; full infer. reports complete SceneFoam inference, including reference generation, four color-fitting steps, and refinement. Dashes mark inapplicable timing columns. Means weight the four scenes equally; best image-quality scores are bold within each group.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	PF opt. (s)	Full infer. (s)
Visitor information room					
PF, original init.	19.8823	0.6240	0.4650	2292.24	—
PF, learned init. + virtual refs.	23.5840	0.7520	0.2951	2189.85	—
SceneFoam	**24.9066**	**0.8144**	**0.2010**	—	9.21
Bookshelf interior					
PF, original init.	14.3830	0.4411	0.5795	2934.51	—
PF, learned init. + virtual refs.	24.2749	0.8156	0.2707	2228.81	—
SceneFoam	**25.2516**	**0.8459**	**0.2568**	—	9.44
Pavilion and skyline					
PF, original init.	15.1074	0.5192	0.5093	3184.71	—
PF, learned init. + virtual refs.	29.6388	0.9369	0.1321	2641.65	—
SceneFoam	**30.0440**	**0.9440**	**0.1207**	—	9.35
Garden and timber shed					
PF, original init.	17.2470	0.3717	0.4635	2882.20	—
PF, learned init. + virtual refs.	23.3989	0.7082	0.2528	2394.43	—
SceneFoam	**24.3581**	**0.7579**	**0.2214**	—	8.94
Four-scene mean					
PF, original init.	16.6549	0.4890	0.5043	2823.415	—
PF, learned init. + virtual refs.	25.2242	0.8032	0.2377	2363.685	—
SceneFoam	**26.1401**	**0.8406**	**0.2000**	—	9.235

rather than a parameter-preserving export of the Gaussian scene. The learned-initialization-plus-reference baseline evaluates this route with 30,000 PF updates. SceneFoam trains its frontend and native initializer jointly through the PF renderer and learns geometry-first refinement across scenes. Its fixed-weight construction recovers the native structure, aligns colors in four fitting steps, and refines the scene using refreshed source evidence. This provides seconds-scale sparse-view acquisition of the same native representation with higher held-out quality.

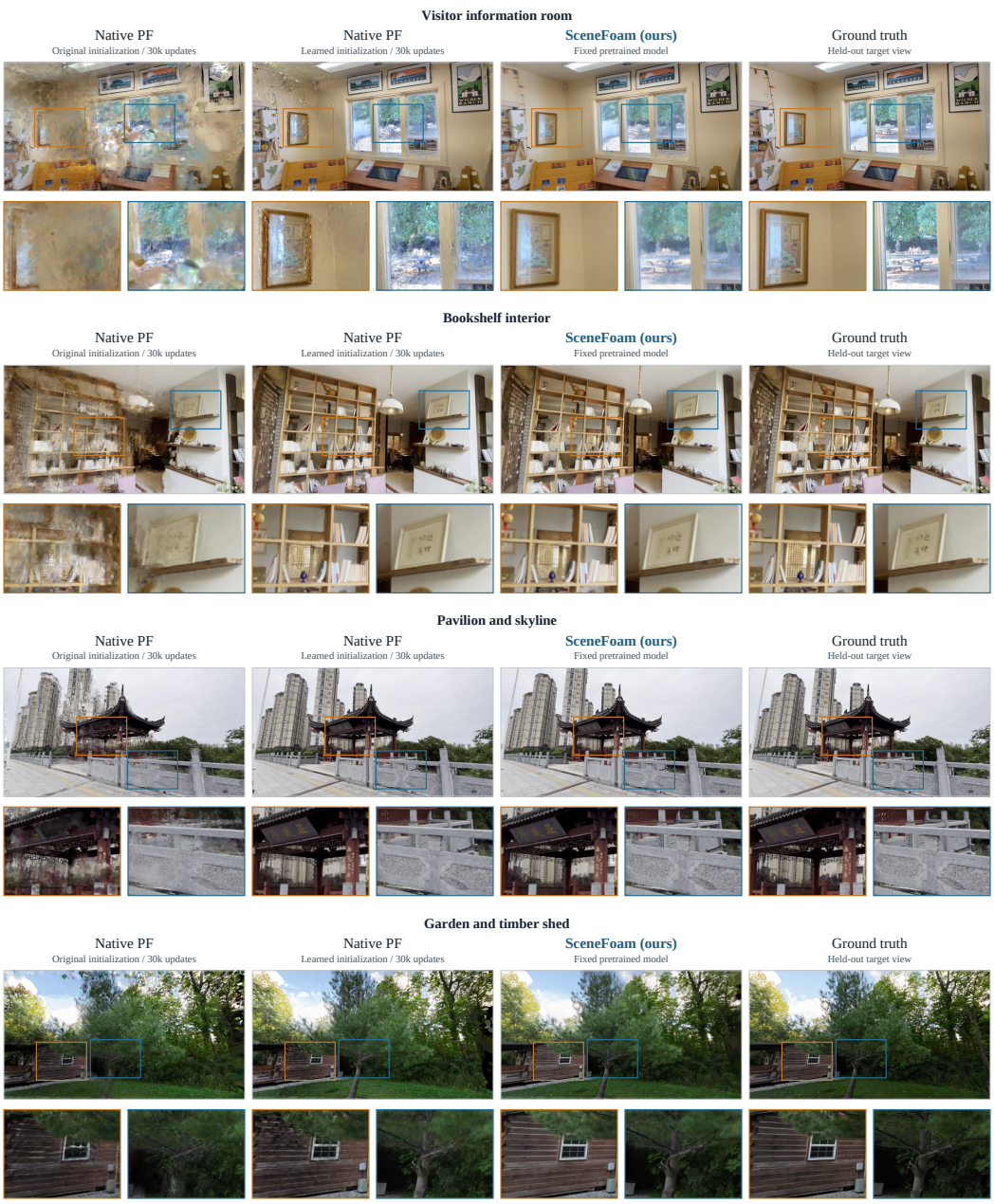

Figure 7: **Native PF acquisition across four scenes.** Top to bottom: visitor information room, bookshelf interior, pavilion and skyline, and garden and timber shed. Columns show original-initialization PF, PF with learned initialization and virtual RGB references, SceneFoam, and ground truth. Table 11 reports the corresponding quality and times. Orange and blue boxes identify matching enlarged regions, highlighting interior surfaces, frame and shelf boundaries, pavilion railings, and timber and vegetation detail.

F REFERENCE IMAGES AND SUPPLEMENTARY VISUAL RESULTS

This appendix compares reference-image sources at a fixed color-fitting budget and provides additional illustrations of scene refinement and reconstruction quality.

F.1 REFERENCE IMAGES AT A FIXED FITTING BUDGET

The four-step comparison fixes the initial native geometry, directional axes, network weights, target cameras, and refinement network. Only the RGB targets used for color fitting change. Source-only fitting uses the input photographs. Adapted-reference fitting adds images rendered by the frontend trained for foam reconstruction. Independent-reference fitting instead uses images from the separate reference model with fixed pretrained weights. All three settings use all 140 DL3DV140 scenes and 1,120 held-out targets, using the main-model weights from Tables 1 and 3.

Table 12: **Reference-image comparison on all 140 DL3DV140 scenes.** Four fitting steps; the main-model weights, initial geometry, axes, and 1,120 target images are fixed. Both PSNR columns and LPIPS use held-out targets. The same refinement network follows every fit.

Fitting targets	Post-fit PSNR ↑	Final PSNR ↑	Final LPIPS ↓
Real source RGB	27.54	27.77	0.178
Real + adapted-frontend RGB	27.31	27.42	0.203
Real + independent reference RGB	**27.61**	**27.97**	**0.153**

Separating scene prediction from reference generation. Independent references raise post-fit PSNR from 27.54 to 27.61 dB and final PSNR from 27.77 to 27.97 dB, while reducing final LPIPS from 0.178 to 0.153. With geometry, directional axes, fitting budget, and refinement network fixed, this comparison isolates the reference source. SceneFoam adapts its frontend for native scene prediction while a separate pretrained model supplies the image references used for color fitting.

F.2 ADDITIONAL ILLUSTRATIONS AND QUALITATIVE COMPARISONS

Refinement protocol and illustration. The three variants in Table 3B start from the same fitted scene $\mathcal{P}_0$ and use matched training budgets and image losses. The sequential variant without refresh supplies updated geometry to the appearance head but retains the original residuals and source-loss gradients; the full variant recomputes these inputs at the updated scene $\mathcal{P}_g$. Figure 8 illustrates the change in source-ray contributions. Appendix B.3 specifies the decoding and gradient paths.

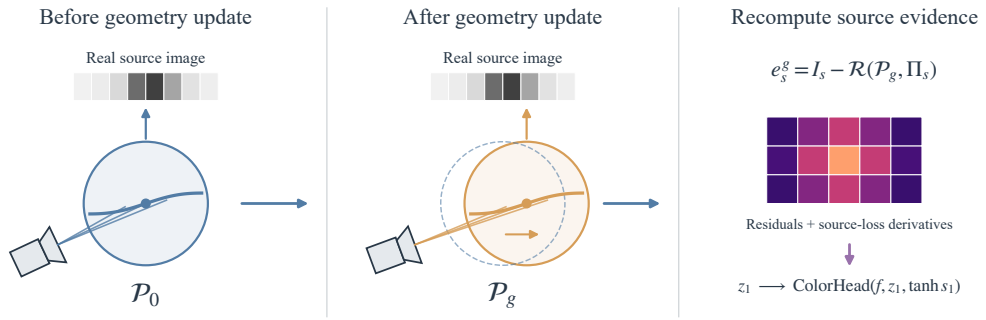

Figure 8: **Why source observations are refreshed.** Schematic bounded-cell cross-sections illustrate how a geometry update changes source-ray contributions. Source images stay fixed. Re-rendering the updated scene $\mathcal{P}_g$ produces new image residuals. Mapping these residuals to texture sites and recomputing source-loss gradients provides evidence for the appearance head under the new geometry. Dashed support denotes the preceding geometry. The pixel strips and heatmap are schematic, not measured errors.

Selection and display. The qualitative set covers a classroom, two different building entrances, a museum, an office, and an indoor garden; the last pair comes from the preselected median/low-PSNR examples. Scene labels identify the examples, and colored boxes mark matching enlarged regions. Error maps use a shared scale and legend. The qualitative and camera-study scenes are distinct from the main teaser, method, and quality panels. Real source images, held-out target images, and representation schematics are labeled separately.



Figure 9: **Extended eight-view reconstruction comparison.** Matched cameras and crops expose differences in thin geometry, directional appearance, texture, and visibility. Every target is excluded from the construction inputs.

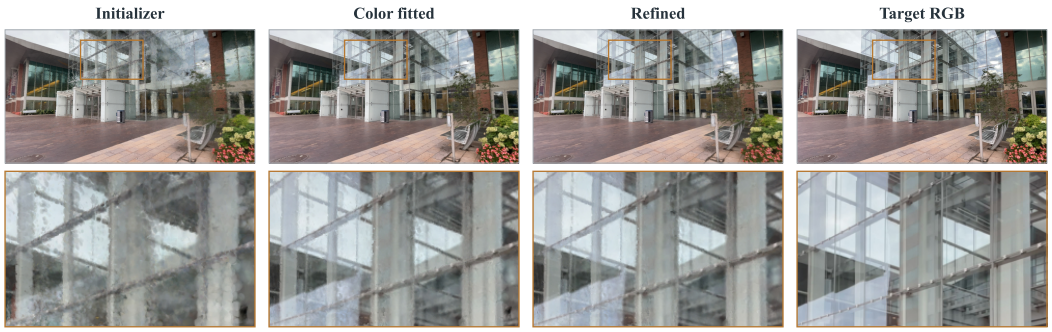

Figure 10: **Native construction snapshots.** A separate entrance scene shows the learned initializer, the scene after directional adaptation and four color-fitting steps, and the final refined scene, followed by target RGB. Full views and marked crops share the same source images and held-out camera.

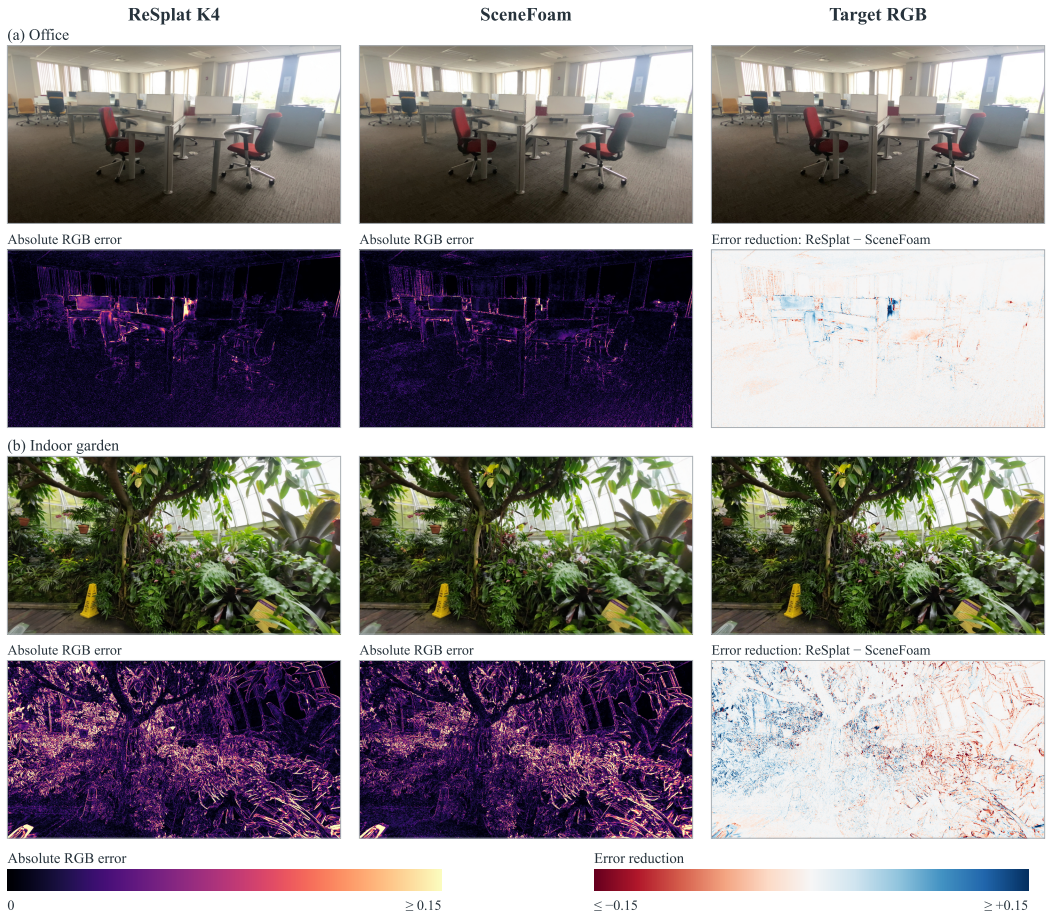

Figure 11: **Spatial distribution of reconstruction error.** An office and indoor garden show full ReSplat, SceneFoam, and target views. Absolute RGB errors share a $[0, 0.15]$ scale; their difference uses $[-0.15, 0.15]$. Blue in the difference image indicates smaller SceneFoam error. The color scales apply to the error visualizations; RGB panels show the rendered images.

G NATIVE-SCENE ACQUISITION AND RESOURCE COSTS

This appendix describes the native scene acquired by SceneFoam, its construction stages, and its reuse for rendering and interaction, together with construction time and field storage.

G.1 RECONSTRUCTION TARGET AND NATIVE FUNCTIONALITY

Sparse images to a native spatial partition. SceneFoam learns to recover a textured Power Foam from sparse calibrated images. The reconstruction target comprises bounded spatial cells, displaced internal surfaces, surface-attached texture, and directional radiance. Power Foam supplies the representation and rendering interfaces (Govindarajan et al., 2026); SceneFoam acquires their native parameters through jointly learned prediction, source-conditioned color fitting, and geometry-first refinement. These stages combine a learned scene prior with image evidence for the current geometry, producing a textured spatial partition for novel-view synthesis and downstream queries.

Representation-specific scene acquisition. 3DGUT retains Gaussian particle parameters while defining image formation through unscented projection and 3D particle responses evaluated along rays (Wu et al., 2025). The standard-Gaussian baselines in our evaluation predict scenes for the conventional 3DGS image-formation model, which evaluates projected 2D responses (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025; 2026). Scene acquisition for 3DGUT therefore calls for reconstruction or adaptation matched to its image-formation model. SceneFoam learns to acquire native Power Foam parameters directly, coupling reconstruction with the cell geometry, local texture, and directional radiance used by its renderers.

Quality and native reuse. Tables 1 and 2 evaluate held-out image quality under the stated sparse-input protocols. SceneFoam improves all three reported metrics over ReSplat across the three evaluation settings. Figures 1 and 5 show how the recovered structure is reused: the same field supports distorted-camera queries, reflected and refracted rays, and interaction on its tetrahedral dual. Appendix H specifies the downstream queries and settings. Together, these experiments evaluate the acquired foam as both an image-synthesis representation and a reusable scene for native rendering and interaction.

G.2 ONE-TIME CONSTRUCTION AND SUBSEQUENT RENDERING

Table 11 compares acquisition into the same native PF representation on four scenes: complete SceneFoam inference takes 8.94–9.44 s, while PF with learned initialization and virtual references uses 36.50–44.03 minutes of optimization. SceneFoam construction combines frontend prediction, seven-image independent reference generation, native initialization, directional conditioning, four color-fitting steps, and geometry-first refinement. Table 13 reports mean and median per-scene construction times over all 140 DL3DV140 scenes on an NVIDIA H800. The resulting native field is stored once and reused across subsequent camera and ray queries.

Table 13: **H800 scene-construction time on DL3DV140.** Mean and median over all 140 scenes for eight-view reconstruction at 512×960. The methods produce different native representations. SceneFoam uses seven midpoint reference images and four color-fitting steps.

Method	Mean construction (s)	Median construction (s)
ReSplat	0.849	0.848
SceneFoam	9.392	9.336

Native construction stages. Each foam cell carries an internal surface, multiple texture sites, and directional color coefficients. The jointly trained frontend and initializer predict this structure from the source images. An independent reference branch supplies RGB observations for directional color fitting, and geometry-first refinement recomputes native rendering evidence after updating the surfaces. Table 13 reports the end-to-end cost of this construction procedure, whose output contains the geometry and appearance consumed by the native renderers.

Construct once, query natively. Network weights remain fixed during scene construction. Four numerical fitting steps align the directional colors with source-conditioned observations, followed by the learned geometry and appearance updates. The final geometry and radiance parameters form a stored native scene. Each rendering backend prepares its acceleration structures over these same parameters (Appendix H.1). Camera-model changes, reflected and refracted rays, and the interaction queries in Appendix H operate on this reconstructed scene. Reconstruction supplies the asset once; subsequent rendering receives the stored field and the query configuration.

G.3 Field storage

For $N = 245{,}760$ cells and $T = K = 8$, the eight core float32 arrays total 390.9375 MiB: 180 MiB for directional RGB coefficients, 180 MiB for raw directional-axis parameters, and 30.9375 MiB for centers, radii, densities, quaternions, texture coordinates, and heights. Shared axes are stored explicitly at each site. Here $1\,\mathrm{MiB} = 2^{20}$ bytes. The breakdown uses all 140 DL3DV140 scenes; file sizes include serialization overhead, whereas array payloads count only stored elements.

Component	Mean per scene (MiB)
Core field-array payload	390.9375
Rasterization adjacency, other fields, and file overhead	14.4042
Rasterization `.pt` file total	**405.3417**
Ray-tracing field `.npz` file	390.9394
Ray-tracing full-power-graph `.npz` file	69.2798
Ray-tracing two-file total	**460.2192**

The `.pt` file stores sphere-overlap adjacency; its 14.4042 MiB remainder is the file size minus the core payload. Ray tracing instead stores full-power-graph adjacency indices, offsets, valid-node flags, and four float32 precomputed values per directed edge. In the separate three-scene camera study, these graph arrays average 69.0103 MiB, including 54.2708 MiB of edge precomputations and excluding archive headers; this is a raw-payload statistic, distinct from the full-DL3DV140 file-size averages above.

H Native rendering and downstream interfaces

This section presents native rasterization and ray tracing, camera-ray queries, optical insertions, and deformable simulation directly on the cell complex's tetrahedral dual. Power Foam provides the rendering interfaces; SceneFoam acquires the textured field from sparse images. Optical properties, elastic parameters, and interaction handles are supplied for each downstream task.

H.1 Native asset and renderer interfaces

SceneFoam's output contains the geometry and radiance parameters consumed by native Power Foam renderers (Govindarajan et al., 2026): cell centers, radii, density, local frames, texture sites and heights, and directional appearance. Rasterization uses a sphere-overlap graph to identify potential cell intersections. Adjacency-walk ray tracing instead uses the full power diagram, with starting cell

$$i_0(o) = \arg\min_i \left(\|o - p_i\|^2 - r_i^2 \right) \tag{32}$$

for ray origin o. A cell with an empty power domain is omitted from traversal. These acceleration structures index the same field parameters.

Non-pinhole queries. A ray-map interface supplies an origin and unit direction for each pixel. Generic cone rasterization groups such rays for spatial culling; cell traversal follows their intersections directly. For a central camera with normalized image-plane radius ρ, pinhole and equidistant directions use polar angles $\theta = \arctan \rho$ and $\theta = \rho$, respectively, with each query's focal scale. This changes the output camera while preserving scene parameters. SceneFoam's reconstruction inputs and four-sample training renderer use posed pinhole views.

Traversal check. The storefront field has 245,760 cells and 3,509,972 directed full-power-graph entries; 7,635 empty-domain cells remain serialized but are omitted from traversal. At 128 sampled pixels per camera, the starting-cell lookup matches exhaustive weighted nearest-cell search. The two-backend image differences are recorded below.

H.2 Backend agreement across scenes

For each test, we reload the stored field and match camera origins, ray directions, backgrounds, precision, sample offsets, and image resolution across the two native backends. The aggregate uses all 140 DL3DV140 scenes, one view per scene, and one ray per pixel. The three-scene camera sweep separately matches both renderers at each camera setting.

Table 14: **Agreement between the two rendering backends.** Native rasterization and ray tracing receive identical one-sample rays at 960×512. The aggregate covers all 140 DL3DV140 scenes, one view each. Ray counts are per backend. MAE denotes mean absolute RGB difference; the dash marks an unreported aggregate maximum.

Scene / set	RGB MAE	RGB maximum	Ray count
Storefront	1.82×10^{-4}	2.94×10^{-3}	491,520
Gym	1.475×10^{-4}	1.118×10^{-3}	491,520
Exhibit hall	1.614×10^{-4}	4.536×10^{-3}	491,520
Brick facade	1.147×10^{-4}	1.340×10^{-3}	491,520
DL3DV140 (140 scenes)	1.479×10^{-4}	—	68,812,800

H.3 Extended rendering and camera studies

The native-query study varies the renderer or camera while holding a constructed field fixed. Power Foam provides the bounded-cell rasterizer and adjacency ray tracer (Govindarajan et al., 2026); the inputs here are fields reconstructed by SceneFoam from sparse observations. Figure 12 organizes additional scenes under this protocol.

Serialized-field protocol. Each scene is constructed once; centers, radii, frames, densities, texture sites, displacements, axes, and RGB coefficients are saved. A fresh rendering process reloads these arrays and compares file checksums before and after the camera tests to verify that scene parameters stay unchanged. Each backend prepares its own topology and acceleration structures before evaluating camera queries. Source RGB and reconstruction networks are used during construction; the rendering process receives the serialized field and the query configuration.

Backend agreement. At each camera, both rendering paths consume the same ray origins, directions, and subpixel offsets. Background, precision, tone mapping, and sample count are matched. For $H \times W$ RGB outputs I_{rast} and I_{ray}, record

$$E_{\mathrm{mean}} = \frac{\|I_{\mathrm{rast}} - I_{\mathrm{ray}}\|_1}{3HW}, \qquad E_{\max} = \|I_{\mathrm{rast}} - I_{\mathrm{ray}}\|_\infty. \tag{33}$$

The error image uses $\frac{1}{3}\sum_c |I_{\mathrm{rast}}(u,v,c) - I_{\mathrm{ray}}(u,v,c)|$ with one color scale across scenes. Reconstruction PSNR and LPIPS use ground-truth target images in the reconstruction benchmark; Eq. (33) measures agreement between two evaluations of the same field.

H.3.1 Camera ray maps and distortion settings

Let R rotate camera coordinates to world coordinates and o be the camera center. The ray at pixel (u, v) is $o + tR\hat{d}(u, v)$, with unit camera-space direction $\hat{d}$. Coordinates include the declared subpixel offset. We use the pinhole and distortion conventions of the OpenCV calibration model (OpenCV, 2026a).

Pinhole. For focal lengths f_x, f_y and principal point (c_x, c_y),

$$q_x = (u - c_x)/f_x, \quad q_y = (v - c_y)/f_y, \quad \hat{d}_{\mathrm{pin}} = \frac{(q_x, q_y, 1)}{\|(q_x, q_y, 1)\|_2}. \tag{34}$$



Figure 12: **Fixed-field rendering across scenes.** Columns change the camera model; each row uses one reconstructed scene at a fixed camera pose. The rows show a gym, an exhibit hall and a brick facade, all separate from the other qualitative figures. The columns show a 90° pinhole camera, a 100° equidistant camera and the mild radial setting, respectively. The third row uses the same central 600×320 crop in every column; numerical measurements use the full images.

Equidistant fisheye. With one focal scale f, let $\rho = \sqrt{(u - c_x)^2 + (v - c_y)^2}$ and $\theta = \rho/f$. The zero-distortion equidistant model (OpenCV, 2026b) gives

$$\hat{d}_{\text{fish}} = \left(\sin\theta \frac{u - c_x}{\rho},\ \sin\theta \frac{v - c_y}{\rho},\ \cos\theta \right), \tag{35}$$

with $\hat{d} = (0, 0, 1)$ at $\rho = 0$. The extended sweep defines field of view horizontally: $f = W/\text{FoV}_{\text{rad}}$. It uses 100°, 120°, and 150° at the same pose.

Radial and tangential distortion. For undistorted normalized coordinates (x, y), write $s = x^2 + y^2$ and $a = 1 + k_1 s + k_2 s^2$. The pixel-to-ray operation numerically inverts

$$\begin{aligned} x_d &= xa + 2p_1 xy + p_2(s + 2x^2), \\ y_d &= ya + p_1(s + 2y^2) + 2p_2 xy, \end{aligned} \tag{36}$$

to match $x_d = (u - c_x)/f_x$ and $y_d = (v - c_y)/f_y$. Pixel-center coordinates and inversion are computed on the CPU in NumPy float64, using exactly 20 Newton iterations without early stopping or a separate polishing step. A pixel is valid when the result is finite and $r = \|D(x, y) - (x_d, y_d)\|_2 < 10^{-6}$, where D is the forward map in Eq. (36). World-space ray directions are then formed and normalized, and the rays are cast to float32 for rendering.

Camera settings. All queries use 960×512 pixels with centered principal point. The pinhole reference has 90° horizontal field of view; fisheye queries use 100°, 120°, and 150°. Mild and larger radial distortion use $(k_1, k_2) = (-0.10, 0.02)$ and $(-0.20, 0.02)$, respectively. The radial–tangential query adds $(p_1, p_2) = (0.001, -0.001)$ to the mild radial setting. The camera-query measurements use one ray per pixel.

Measurement procedure. The sweep covers all 960×512 pixels in each of three fixed fields, with a $(0.5, 0.5)$ pixel-center offset. Table 15 reports the maximum r recomputed by forward distortion of

the recovered float64 coordinates, before casting rays to float32. This residual quantifies the FP64 camera-map inversion before conversion to float32 rendering rays. Valid fraction measures numerical ray-map validity; reconstruction quality is evaluated against held-out RGB in Tables 1 and 2.

Table 15: **Camera-query measurements on three fixed fields.** One ray per pixel, 960×512. The FP64 inversion residual is $\max r$ over all pixels and scenes before float32 ray conversion (distorted queries only). All queries preserve the serialized field parameters.

Query	Valid (%)	FP64 inversion residual	Unchanged scenes
Pinhole	100.0	—	3/3
Fisheye 100°	100.0	—	3/3
Fisheye 120°	100.0	—	3/3
Fisheye 150°	100.0	—	3/3
Radial, mild	100.0	2.483×10^{-16}	3/3
Radial, larger	100.0	5.796×10^{-16}	3/3
Radial + tangential	100.0	3.511×10^{-16}	3/3

H.4 Secondary-ray examples

The main examples use an inserted planar mirror and a dielectric sphere in fixed reconstructed fields (Figure 5). The mirror reflectance is 0.96. The glass index is 1.5 in air, with two interfaces and Schlick Fresnel weights (Schlick, 1994), without higher internal bounces. The inserted geometry and optical properties are supplied; the surrounding foam's geometry and directional coefficients stay fixed. Native ray tracing queries the reconstructed field along the reflected or refracted paths (Govindarajan et al., 2026).

For incident direction d and surface normal n, reflection uses $d_r = d - 2(d^\top n)n$. Refraction uses Snell's law at the supplied interfaces. These examples demonstrate native secondary-ray access to the same sparse-view reconstruction; the displayed object transforms, camera, and rendering settings define each query.

Ray-segment convention and site-based appearance. Equation (10) evaluates directional appearance at a site's base-plane position. For each ray segment, let o be its origin, u its unit propagation direction, and $x = o + \ell u$, $\ell > 0$, a query point in the foam. A primary ray uses the camera center. A reflected or refracted segment uses its most recent interface point, displaced by the numerical ray-origin offset; after traversing both glass interfaces, this is the exit point. The shader uses $\widetilde{d}_{it} = (x_{it} - o)/\|x_{it} - o\|$, with the same propagation-direction sign convention as u, rather than retaining the original camera center. This preserves the site's directional-color evaluation used in native camera rendering. Spatial interpolation then combines these site colors at the query point. The convention approximates direction-dependent appearance; ray–cell intersections, segment lengths, and the reflection and refraction directions are computed from the actual ray.

Conditional angular approximation bound. Write $x_{it} = x + \delta_{it}$, $\delta_\parallel = u^\top \delta_{it}$, and $\delta_\perp = \delta_{it} - \delta_\parallel u$. If $\ell + \delta_\parallel > 0$, the angular difference between site and ray directions is exactly

$$\theta_{it} = \operatorname{atan2}(\|\delta_\perp\|, \ell + \delta_\parallel).$$

In particular, if $\|\delta_{it}\| \le \rho_{it} < \ell$, then

$$\theta_{it} \le \arcsin(\rho_{it}/\ell), \qquad \|\widetilde{d}_{it} - u\| \le 2\sin\big(\tfrac{1}{2}\arcsin(\rho_{it}/\ell)\big)\,. \tag{37}$$

To see this, the vectors $\ell u + \delta_{it}$ lie in a ball of radius ρ_{it} centered at ℓu. The cone tangent to this ball has half-angle $\arcsin(\rho_{it}/\ell)$. The chord-distance bound follows from the angle between unit vectors. Thus the controlling quantity is the site-to-query separation relative to the current segment length, rather than the camera field of view alone.

Propagation to directional color. For one site and RGB channel, let $\Delta C = \max_k C_k - \min_k C_k$ and $\tau_{\max} = \max_k \tau_k$. Suppose the directional denominator floor in Eq. (10) is inactive along the straight interpolation between u and $\widetilde{d}_{it}$. The normalized exponential weights then give a Lipschitz

color function with constant $L \le \tau_{\max}\Delta C$. Indeed, for logits $l_k(d) = -\tau_k\|d - a_k\|$ and their normalized weights p_k, differentiating $f(d) = \sum_k p_k C_k$ gives $\nabla f = \sum_k p_k(C_k - f)\nabla l_k$. Since $\|\nabla l_k\| \le \tau_{\max}$ and $|C_k - f| \le \Delta C$, this establishes the bound; the lower color clamp is nonexpansive. Continuity extends the argument through the isolated nondifferentiable points of the distance function. Consequently, channelwise,

$$|c_{it}(\widetilde{d}_{it}) - c_{it}(u)| \le 2\tau_{\max}\Delta C \, \sin\!\big(\tfrac{1}{2}\arcsin(\rho_{it}/\ell)\big)\,. \tag{38}$$

Spatial weights in Eq. (9) are nonnegative with sum at most one, as are the volume-compositing weights for a fixed ray. Holding geometry, visibility, and any culling decisions fixed therefore bounds the pixel-color difference by the corresponding weighted sum of site errors. The estimate applies when both evaluations use the directional-color branch; a change to a constant frustum fallback is a separate case. Small site-to-distance ratios and smooth directional appearance tighten this local approximation. Equations (37) and (38) quantify the difference between evaluating the same fitted field with site-based and propagation directions under the stated conditions.

H.5 Deformable simulation on the native tetrahedral dual

We simulated the pull-and-release interactions in Figures 1 and 13 directly on each scene's native tetrahedral dual, with selected cell centers as mechanical nodes. The extended position-based dynamics (XPBD) and finite element method (FEM) updates used scene-specific tetrahedra derived from the weighted power diagram and the same transport rule for native foam parameters. No separately constructed mechanical proxy or rest-space binding was used. The interacting solid regions, material properties, constraints, and handle trajectories were user-specified; these define the elastic interaction applied to each reconstructed radiance field.

H.5.1 From the cell complex to simulation degrees of freedom

Weighted dual and bounded support. At rest, form the regular triangulation of the reconstructed sites $\{(p_i, r_i^2)\}$, the dual of the complete weighted power diagram. Ordinary Delaunay is its equal-weight special case. The sphere-clipped cells in Eq. (1) instead define the bounded structure associated with the weighted α-complex (Govindarajan et al., 2026). These are distinct objects: the complete regular triangulation supplies candidate tetrahedra, while the bounded-support and solid-domain tests below select the mechanical volume. Auxiliary empty-space sites used solely to accelerate ray traversal do not enter the mechanical construction.

Solid selection and admissible elements. Let Ω be a supplied closed solid selection in rest coordinates, represented by an oriented, watertight boundary. For each nondegenerate regular tetrahedron e with vertices $p_{i_0}, \dots, p_{i_3}$, solve for its power center q_e from

$$2(p_{i_j} - p_{i_0})^\top q_e = \|p_{i_j}\|^2 - r_{i_j}^2 - \|p_{i_0}\|^2 + r_{i_0}^2, \qquad j = 1, 2, 3.$$

Retain only elements whose common power value $\|q_e - p_{i_0}\|^2 - r_{i_0}^2 \le 0$, whose entire closed tetrahedron is contained in Ω, and whose dimensionless quality

$$Q_e = \frac{12(3V_e)^{2/3}}{\sum_{a<b,\ a,b\in e}\|X_a - X_b\|^2} \quad \text{satisfies} \quad Q_e \ge Q_{\min} > 0.$$

Here V_e is the positive rest volume; we used the mesh-quality threshold $Q_{\min} = 0.05$. Full-element containment uses tetrahedron–boundary intersection and inside/outside classification, rather than a centroid-only test, so connections crossing excluded cavities are rejected. Orient retained elements positively, collect their faces and edges, and retain the face-connected component containing the interaction seed. An admissible mechanical domain has a nonempty tetrahedral interior and a manifold boundary; orphan vertices and lower-dimensional branches are excluded. Hidden weighted sites with no incident retained tetrahedron are not mechanical nodes. Thus the mechanical volume is a selected subcomplex, not the convex hull of the complete scene or an automatic identification of radiance support with solid material.

Let $\mathcal{M} = (X, \mathcal{T})$ denote this rest domain, with $X_a = p_{i(a)}$. The solver updates $x_a(t)$ and transfers the center directly:

$$p_{i(a)}(t) = x_a(t). \tag{39}$$

All cells outside the active vertex set remain fixed. For $e = (a_0, a_1, a_2, a_3)$, define

$$\begin{aligned} D_m^e &= [X_{a_1} - X_{a_0}, X_{a_2} - X_{a_0}, X_{a_3} - X_{a_0}], \\ D_s^e(t) &= [x_{a_1} - x_{a_0}, x_{a_2} - x_{a_0}, x_{a_3} - x_{a_0}], \\ F_e(t) &= D_s^e(t)(D_m^e)^{-1}, \qquad V_e = \det(D_m^e)/6. \end{aligned} \tag{40}$$

The rest connectivity $\mathcal{T}$ and matrices D_m^e stay fixed during a simulation. Their deformed tetrahedra need not remain Delaunay. Rendering connectivity, in contrast, is rebuilt from the transported foam and is not used to redefine the material's rest state.

H.5.2 XPBD AND FEM UPDATES

Masses, constraints, and handles. We assigned lumped nodal masses $m_a = \rho_{\text{mass}} \sum_{e \ni a} V_e/4$, with material mass density ρ_{mass} distinct from optical density σ_i. Along the selected region's longest rest-space axis, boundary nodes in the bottom 10% were fixed as anchors and those in the top 10% formed the handle set. Anchors satisfied $x_a = X_a$, while handles followed a user-specified lateral trajectory during loading. Releasing the handles allowed their motion to evolve under the elastic solver with the anchors fixed. Constraint velocities followed the handle trajectory during loading.

Position-based update. Extended position-based dynamics (XPBD) (Macklin et al., 2016; Müller et al., 2007) uses edge constraints $\mathcal{C}_{ab} = \|x_a - x_b\| - \|X_a - X_b\|$ and relative signed-volume constraints $\mathcal{C}_e = \det F_e - 1$. For a constraint $\mathcal{C}(x)$ with compliance α, inverse masses w_a, and substep Δt, the update is

$$\Delta\lambda = \frac{-\mathcal{C}(x) - \widetilde{\alpha}\lambda}{\sum_a w_a \|\nabla_{x_a}\mathcal{C}\|^2 + \widetilde{\alpha}}, \qquad \Delta x_a = w_a \nabla_{x_a}\mathcal{C}\,\Delta\lambda, \qquad \widetilde{\alpha} = \alpha/(\Delta t)^2. \tag{41}$$

Predict free positions from current velocities and external forces, impose the kinematic constraints, and iterate the edge and volume projections in a fixed order. Safeguard the prediction and the prescribed boundary path against element inversion before projection. Multipliers start at zero each substep and accumulate within it; locked nodes have zero inverse mass, and constraints with no free node are skipped. An orientation-preserving line search scales a proposed correction and its multiplier increment together so every intermediate tetrahedron satisfies $\det F_e \geq J_{\min} > 0$. The accepted position difference divided by Δt gives the free-node velocity. A substep with an infeasible prescribed handle displacement is subdivided before acceptance; the interaction stops if no admissible substep can be found above the supplied minimum step.

Finite-element update. Linear tetrahedral kinematics and compressible neo-Hookean elasticity (Sifakis & Barbič, 2012) give, for $J_e = \det F_e > 0$ and supplied Lamé parameters μ, λ,

$$U(x) = \sum_e V_e \left[\frac{\mu}{2}(\|F_e\|_F^2 - 3) - \mu \log J_e + \frac{\lambda}{2}(\log J_e)^2 \right], \tag{42}$$

$$P_e = \mu(F_e - F_e^{-T}) + \lambda \log J_e\, F_e^{-T}. \tag{43}$$

The three non-origin nodal forces are the columns of $-V_e P_e (D_m^e)^{-T}$; the origin-node force is minus their sum. A backward-Euler step is defined by the constrained incremental potential

$$x^{n+1} = \underset{x \text{ admissible}}{\overset{\text{local}}{\arg\min}} \frac{\|x - x^n - \Delta t\, v^n\|_M^2}{2(\Delta t)^2} + U(x) - (f_{\text{ext}}^{n+1})^\top x, \qquad v^{n+1} = \frac{x^{n+1} - x^n}{\Delta t}.$$

Here M is the lumped mass matrix and external force is fixed during the substep. Admissibility includes the prescribed boundary positions and $J_e \geq J_{\min}$. A descent method with energy-decreasing, orientation-preserving backtracking solves the incremental problem to the supplied tolerance. The line search stays in the positive-determinant component throughout each trial path; $\log J_e$ is never evaluated on an inverted element. If a feasible converged substep cannot be obtained, reduce the substep instead of accepting an inversion. The resulting update is a local implicit solution of the nonconvex elastic energy.

H.5.3 TRANSPORT OF THE NATIVE SURFACE AND APPEARANCE

One transport map per cell. The node position in Eq. (39) is unique, but its incident tetrahedra generally have different F_e. Define rest-volume weights $\omega_{ie} = V_e / \sum_{f \ni i} V_f$ on those tetrahedra. For each orientation-preserving polar decomposition $F_e = R_e S_e$, let q_e be a unit quaternion for R_e. Align quaternion signs to the incident element of largest weight, breaking ties by element index, and set

$$q_i = \frac{\sum_{e \ni i} \omega_{ie} q_e}{\| \sum_{e \ni i} \omega_{ie} q_e \|}, \qquad S_i = \exp\left(\sum_{e \ni i} \omega_{ie} \log S_e \right), \qquad F_i = R(q_i) S_i. \tag{44}$$

The reference quaternion has positive weight, so sign alignment makes its dot product with the weighted sum strictly positive. The normalization is therefore defined. Moreover, S_i is positive definite and $R(q_i)$ is a proper rotation, giving $\det F_i > 0$. This local transport rule reproduces a uniform affine deformation and avoids assigning different frames to a shared cell. It interpolates the incident element deformations into one cell-level transport map. In the following equations, $F = F_i$ for cell i.

With rest local frame (n_i, t_i, b_i), transport each surface-attached site offset by

$$d_{it} = r_i(u_{it} t_i + v_{it} b_i) + h_{it} n_i, \qquad d'_{it} = F_i d_{it}. \tag{45}$$

The normal follows the inverse transpose, and the tangent is re-orthogonalized:

$$n'_i = \frac{F_i^{-T} n_i}{\| F_i^{-T} n_i \|}, \qquad t'_i = \mathrm{normalize}\big(F_i t_i - n'_i (n_i'^T F_i t_i)\big), \qquad b'_i = n'_i \times t'_i. \tag{46}$$

Store the frame in the renderer's quaternion convention. With volume-equivalent scalar support radius $r'_i = r_i (\det F_i)^{1/3}$, re-encode the transformed detail as

$$h'_{it} = n_i'^T d'_{it}, \qquad u'_{it} = t_i'^T d'_{it} / r'_i, \qquad v'_{it} = b_i'^T d'_{it} / r'_i. \tag{47}$$

All quantities on the right are evaluated from the rest field and the current deformation, avoiding accumulated re-encoding drift. Directional axes rotate as $a'_{itk} = R(q_i) a_{itk}$; RGB coefficients, concentrations, and optical density remain attached to the cell. The scalar radius is a volume-equivalent approximation to an anisotropically deformed support. Reconstructing the power diagram from p'_i, r'_i consequently defines a new native foam rather than the exact affine image of every original cell. Rebuild rendering adjacency before rendering, while retaining the fixed mechanical connectivity. These operations transport a textured radiance field under prescribed elastic interaction without changing the reconstruction-network weights.

H.5.4 SIMULATION SETTINGS AND RENDERED INTERACTION

Scenes and rollout. Figure 13 shows XPBD on the classical-statue scene (top) and FEM on the courtyard-bin scene (bottom). Each simulation used the selected native dual of its own reconstructed scene. Both followed the loading and release protocol above over $0 \le t \le 1$ s, with nominal step size $\Delta t = 1/600$ s and 600 nominal intervals. The four columns were rendered at $t = 0$, 0.25, 0.5, and 1 s, corresponding to nominal step indices 0, 150, 300, and 600. The first 300 intervals applied the pull; the remaining 300 evolved the released object. Inversion-safeguard subdivisions, when needed, remained within these intervals and did not change the output times. Table 16 gives the settings for these rendered sequences.

From simulation states to figure panels. The four columns in Figure 13 show the rest state, loading, the end of loading, and the post-release response. Figure 1(e) illustrates the corresponding rest–pull–release stages with XPBD, the extended PBD variant used here. At each output time, Eq. (39) transferred the simulated nodes directly to cell centers, and Eq. (44) transported surface and appearance parameters. We kept mechanical connectivity fixed and rebuilt rendering adjacency from the transported foam. Each row used a fixed camera and display crop; only the selected object was deformed, while the surrounding scene remained fixed.

Table 16: **Native-dual interaction settings.** XPBD and FEM operate on their respective scene-specific duals. The temporal settings correspond to the four panels in each row of Figure 13.

Setting	Value
Mechanical discretization	Each scene's selected native tetrahedral dual; cell centers are nodes
Dual weights	$w_i = r_i^2$, using model-decoded radii
Bounded-support filter	Common power value at the tetrahedron power center ≤ 0
Element-quality threshold	$Q_e \geq 0.05$
Orientation safeguard	$\det F_e \geq 0.05$
Nominal time step	$1/600$ s
Simulation duration	1 s (600 nominal intervals)
Rendered times / step indices	$0,\ 0.25,\ 0.5,\ 1$ s / $0,\ 150,\ 300,\ 600$
Anchors / handles	Bottom 10% fixed / top 10% pulled during loading
Loading interval	0–0.5 s; user-specified lateral handle motion
Release and response	Release after the 0.5 s frame; free handles until 1 s, anchors fixed
XPBD edge compliance	2×10^{-7}
XPBD relative-volume compliance	2×10^{-5}
XPBD constraint sweeps	5 per solver substep
FEM material parameters	$E = 20{,}000$, $\nu = 0.3$, $\rho_{\text{mass}} = 1{,}000$
FEM integration	Backward Euler
FEM normalized-gradient tolerance	10^{-7}
FEM iteration cap	300 per implicit substep



Figure 13: **Direct-dual pull-and-release simulation and native foam rendering.** Top: XPBD on the classical-statue scene. Bottom: FEM on the courtyard-bin scene. Each solver acted directly on its scene's selected native tetrahedral dual. Columns show 0 s (rest), 0.25 s (loading), 0.5 s (end of loading), and 1 s (post-release response). Cameras and crops are fixed within each row. Table 16 specifies the shared timing and solver settings.

I EXPANDED RELATED WORK AND CONTRIBUTION CONTEXT

SceneFoam connects learned scene acquisition with the native Power Foam renderer. The following comparison separates three design choices: the represented scene, the evidence used to acquire or update it, and the ordering of geometry and appearance updates.

I.1 IMAGE-CONDITIONED REPRESENTATIONS AND LEARNED ACQUISITION

Continuous fields and Gaussian prediction. pixelNeRF conditions a continuous radiance field on image features, IBRNet aggregates source-view information along target rays, and MVSNeRF constructs a radiance encoding from plane-swept stereo features (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021). Gaussian predictors make the scene explicit: pixelSplat samples from learned depth distributions (Charatan et al., 2024), MVSplat and MVSGaussian use multi-view matching to locate primitives (Chen et al., 2024a; Liu et al., 2024), and DepthSplat combines monocular depth features with multi-view evidence (Xu et al., 2025). GS-LRM uses transformer-based Gaussian prediction, Long-LRM extends reconstruction to longer image sequences, and Splatt3R combines geometry estimation with Gaussian prediction from uncalibrated image pairs (Zhang et al., 2024; Chen et al., 2025b; Smart et al., 2024). These works develop image-to-scene priors. SceneFoam trains its reconstruction frontend and native initializer together so that the prior predicts bounded supports, internal surfaces, and local directional texture through the PF renderer.

Alternative native primitives. The underlying image formation depends on the representation. NeRF integrates a continuous density and color field; 3DGS composites projected anisotropic particles; and 2DGS intersects oriented planar Gaussian disks (Mildenhall et al., 2020; Kerbl et al., 2023; Huang et al., 2024). FTSplat learns pixel-aligned triangle geometry and connectivity from calibrated views (Xiong et al., 2026). Radiant Foam uses differentiable ray traversal through a spatial partition, while Power Foam adds bounded cells, internal surfaces, local texture, and a rasterization path for the same native field (Govindarajan et al., 2025; 2026). SceneFoam acquires this cell-and-texture structure from sparse observations. Its learned updates operate in the native geometry and appearance coordinates detailed in Appendix B.

I.2 RENDERING FEEDBACK AND GEOMETRY-CONSISTENT APPEARANCE

What rendering feedback supplies. DeepView learns gradient-descent updates for multiplane images, and G3R lifts rendering-loss gradients into a neural-Gaussian scene representation (Flynn et al., 2019; Chen et al., 2024c). ReSplat lifts image rendering errors to a compact Gaussian scene and predicts recurrent parameter and hidden-state updates (Xu et al., 2026); iSplat develops iterative Gaussian refinement (Wu et al., 2026). SplatFormer applies a point transformer to an existing Gaussian reconstruction (Chen et al., 2025a). QuickSplat combines learned initialization with gradient-guided refinement and densification for surface reconstruction (Liu et al., 2025b). These methods establish rendering evidence as an input to learned scene updates.

The foam-specific dependency. A PF cell contains multiple surface-attached texture sites. Cell positions, radii, orientations, and displacements determine the ray intervals, visibility, and interpolation weights through which these sites contribute to pixels. In the affine image-formation model of Eq. (2), a geometry update changes the operator acting on directional color coefficients. SceneFoam follows this dependency at two points: it fits colors with geometry and the directional basis fixed, then recomputes texture-site residuals and source-loss gradients after the refiner changes geometry. The appearance head consumes the refreshed evidence and the geometry update within the same refinement block. This mechanism is especially pertinent where geometry changes visibility, occupied ray segments, or texture interpolation, because those changes alter the observations assigned to a site.

Experimental connection. Table 3 compares the reported simultaneous, sequential-with-retained-evidence, and geometry-first-refreshed variants from a common fitted scene. It also measures color fitting before refinement. Table 4 evaluates source-conditioned directional bases, and Table 12 compares sources of fitting supervision. Together, these studies connect the native image-formation dependencies to the stages that acquire the final foam. The four-scene native-PF results in Table 11 evaluate

the complete learned acquisition against original-initialization fitting and learned-initialization fitting with virtual references.

I.3 Reference supervision, scene fitting, and renderer reuse

Using sparse and synthesized observations. RegNeRF regularizes geometry and appearance in sparse-view NeRF fitting, including patches rendered from unobserved viewpoints (Niemeyer et al., 2022). FreeNeRF regularizes input frequency and near-camera density for few-shot fitting (Yang et al., 2023). MVSplat360 couples Gaussian features with video diffusion to synthesize novel views (Chen et al., 2024b). SceneFoam uses a separate reference branch to provide RGB fitting targets at source-derived cameras. Those targets supervise the native directional coefficients together with real photographs; the geometry-first refiner then predicts the final field. Appendix E.2 measures the longer native fitting route supplied with learned initialization and virtual RGB references, while Appendix F.1 compares reference sources at four fitting steps.

Acquiring the field used by the renderer. 3DGRT enables ray tracing of Gaussian particles, 3DGUT uses unscented projection and three-dimensional particle responses for distorted cameras and secondary rays, and UTrice studies rasterization and tracing through triangle primitives (Moënne-Loccoz et al., 2024; Wu et al., 2025; Liu et al., 2025a). These methods specify how scene parameters produce images. Power Foam supplies SceneFoam's native rendering interface (Govindarajan et al., 2026); our contribution is learning to obtain the field consumed by that interface. The output is a stored spatial partition whose same geometry and radiance serve rasterization, adjacency-walk ray tracing, and the downstream queries of Appendix H. This separates the learned acquisition mechanism, its reconstruction-quality evidence, and the native operations enabled by the acquired scene.