RevTet: Adaptive and Reversible Tetrahedral Neural Representations

Anonymous authors
Paper under double-blind review

Abstract

Novel-view synthesis enables immersive exploration of scenes reconstructed from images, making radiance fields an important representation for visual computing. However, directly learned volumetric attributes remain difficult to combine with movable geometric support and structurally reversible resolution changes on a conforming mesh. We introduce RevTet, a tetrahedral radiance representation that couples constrained vertex optimization with compatible newest-vertex bisection, explicit attribute transfer, and validity-checked inverse events for post-training capacity recovery and levels of detail. Experiments show that importance-guided refinement and vertex motion improve reconstruction in controlled single-scene continuations, geometric repair improves vulnerable cells at comparable image fidelity, and hierarchical coarsening retains much of the learned image quality at reduced cell counts without retraining. These results support hierarchical tetrahedral adaptation as a means of controlling representation capacity while retaining direct field parameters and explicit connectivity, with lossy coarsening providing a tunable quality–size trade-off.

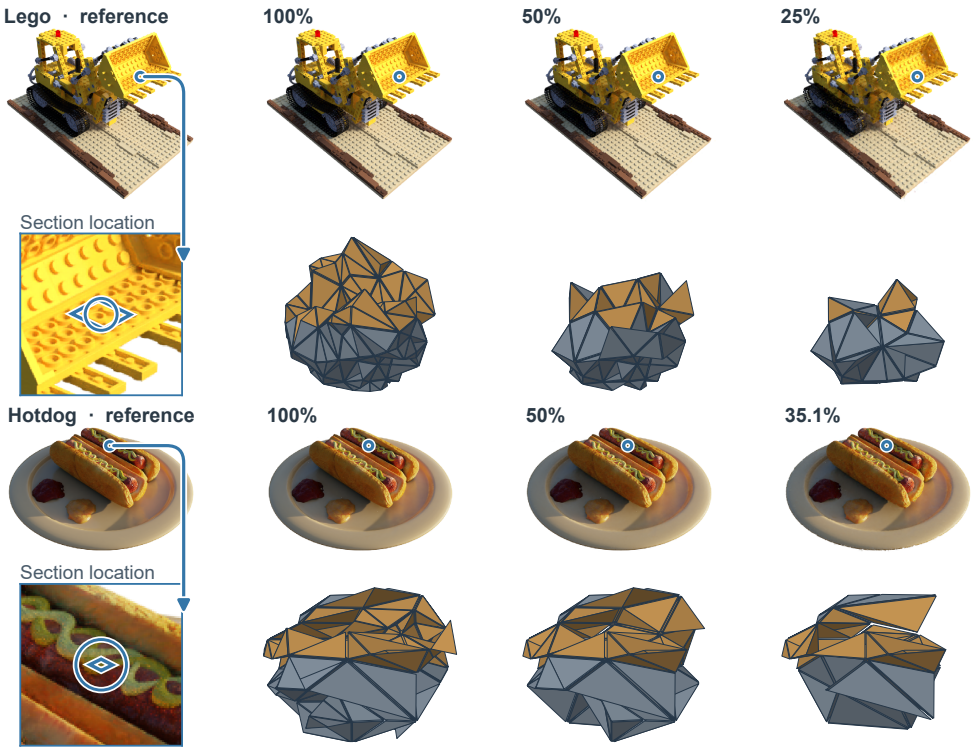

Figure 1: Lego and Hotdog at multiple levels of detail: reference/render pairs, magnified regions, and tetrahedral cutaways. Ochre marks faces near the opening, gray the remainder; colors give spatial context, and percentages are retained tetrahedra.

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

Reconstructing visual scenes from images is central to immersive visualization and the creation of digital environments. Novel-view synthesis addresses this goal by predicting how a scene appears from unobserved viewpoints. Neural radiance fields learn density and directional appearance from posed images (Mildenhall et al., 2020), while explicit representations such as 3D Gaussian Splatting offer a different balance between scene parameterization and rendering (Kerbl et al., 2023). Together, these approaches motivate representations whose spatial capacity can adapt to scene content.

Within radiance-field learning, volumetric discretizations associate density and appearance with spatial cells and define the intervals over which rays accumulate color. Sparse voxels provide structured local support (Sun et al., 2025), whereas Voronoi and tetrahedral meshes allow the support to move with learned geometry (Govindarajan et al., 2025; Mai et al., 2026). Tetrahedra are particularly useful because barycentric interpolation links local attributes to vertex positions and shared faces support cell-to-cell traversal. The challenge is to jointly learn these attributes and positions while allocating a finite number of cells to image detail.

However, little is known about how to combine direct mesh-attribute learning, geometric motion, and structurally reversible capacity allocation within one conforming tetrahedral hierarchy. Radiance Meshes (RM) reports difficulties with directly attached attributes under motion-induced Delaunay connectivity changes and instead queries an auxiliary neural field for cell properties (Mai et al., 2026). This indirect parameterization adds neural-network evaluation and backpropagation to training. Its point-insertion strategy does not provide an explicit inverse refinement hierarchy. Classical compatible bisection supplies refinement rules (Maubach, 1995; Stevenson, 2008), but does not by itself specify how image-driven deformation and attribute transfer should interact. The missing link is a common operator framework that maintains shared-face compatibility, constrains geometric updates, and defines what information survives refinement and coarsening.

We introduce RevTet, which learns explicit radiance attributes on a movable tetrahedral mesh with a persistent refinement hierarchy. Our contributions are threefold: image-guided conforming newest-vertex bisection (NVB) with exact inheritance of the affine coefficient field; constrained vertex learning and field-aware mesh maintenance formulated through a moving-basis identity; and checked inverse events that reclaim capacity and generate offline levels of detail (Fig. 1). These operators avoid training-time Delaunay reconstruction and auxiliary neural attribute queries. Controlled continuations, repair ablations, and quality–budget evaluations establish their complementary roles in reconstruction, mesh quality, and resolution control, while distinguishing reversible connectivity from the learned detail lost during coarsening.

2 Related Work

2.1 Radiance field representations

Neural and explicit volumetric radiance fields commonly use the emission–absorption model (Mildenhall et al., 2020). For a camera ray $r(t) = o + td$ with unit direction d, the pixel color is

$$\begin{aligned} C(r) &= \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\, dt + T(t_f)c_{\text{bg}}, \\ T(t) &= \exp\left[-\int_{t_n}^{t} \sigma(r(s))\, ds\right]. \end{aligned} \tag{1}$$

Here $\sigma \geq 0$ is density, c is directional RGB radiance, T is transmittance, and t_n, t_f bound the integration domain; c_{bg} is the background color. Representations differ in their parameterization and spatial discretization of these quantities. Appendix D.1 gives discrete alpha compositing. Splatting methods also use alpha compositing, but their projected opacities need not equal the exact ray-interval integrals of this model.

NeRF represents density and view-dependent radiance with a coordinate-conditioned neural network and synthesizes images by volume rendering (Mildenhall et al., 2020). Instant-NGP accelerates

neural-field learning through multiresolution hash encodings (Müller et al., 2022). In contrast, 3D Gaussian Splatting (3DGS) directly optimizes anisotropic Gaussian primitives and interleaves optimization with cloning, splitting, and pruning (Kerbl et al., 2023). Its overlapping primitives support efficient splatting without requiring a conforming partition of the scene volume.

Explicit volumetric representations instead organize the field over spatial cells. SVRaster uses adaptive sparse voxels, directly learns corner densities and cell appearance, and allocates resolution through subdivision and pruning (Sun et al., 2025). Radiant Foam learns a Voronoi tessellation and uses cell adjacency for differentiable ray tracing (Govindarajan et al., 2025). Radiance Meshes adopts Delaunay tetrahedra and obtains their attributes from a neural field to accommodate connectivity changes during vertex optimization (Mai et al., 2026). These methods explore different combinations of explicit parameters, movable spatial support, and cell connectivity.

RevTet combines directly learned attributes, movable vertices, and a persistent refinement/coarsening hierarchy. Table 6 compares these structural choices. The following section explains why a conforming tetrahedral hierarchy supports this combination.

2.2 Volumetric meshing and adaptation

The choice of tessellation determines how geometric motion changes the support of learned attributes. Point-set methods such as Voronoi and Delaunay representations construct cells from site positions; moving sites can change both cell shapes and connectivity (Govindarajan et al., 2025; Mai et al., 2026). This flexibility supports geometric adaptation, but changing connectivity requires updating cell identities, shared-face adjacency, and traversal structures while reassigning or re-querying attributes. These coupled tasks add implementation and maintenance costs. RevTet instead keeps connectivity fixed during vertex learning and changes it through recorded local events. Tetrahedra provide affine cell interpolation, making motion and coefficient transfer explicit without training-time Delaunay reconstruction or an auxiliary attribute-query network.

Hierarchical bisection complements this choice by providing a known relation between coarse and fine cells. Maubach defines compatible bisections through vertex orderings (Maubach, 1995); neighboring splits form a *closure* that matches shared-face subdivisions, with completion bounds under suitable initialization (Stevenson, 2008). We choose this construction because a shared midpoint admits exact inheritance of the parent coefficient field, matching faces support adjacency-based ray traversal, and retained parent–child records identify candidate inverse events. These properties make refinement and capacity recovery compatible with direct field parameters. Reversing an event still requires geometric checks and can discard detail learned after splitting.

A refinement hierarchy alone does not control mesh deformation or preserve a field when its vertices move. Lipman relates local orientation to global injectivity under boundary and topological assumptions (Lipman, 2014), motivating our constrained position updates. ALE methods distinguish changes in a field from changes in the mesh carrying it (Hirt et al., 1974; Donea et al., 2004); this distinction motivates the learning-versus-maintenance formulation in Section 3.2. Unlike the simulation settings of MeshGraphNets and LAMP (Pfaff et al., 2021; Wu et al., 2023), our field is learned from images, and mesh maintenance must account for its learned density and appearance.

3 Method

RevTet represents a radiance field by a tetrahedral complex, learned attributes, and a persistent refinement hierarchy. We distinguish field learning from mesh maintenance to motivate refinement, merging, and repair. Figure 2 illustrates the representation and these operators.

3.1 Discretizing a radiance field on moving tetrahedra

Let $\mathcal{T}$ denote tetrahedral connectivity, $X = (x_i)$ vertex positions, and θ all learned attributes, including nodal coefficient vectors θ_i and any cell-local parameters. For a cell $K = (i_0, i_1, i_2, i_3)$, define the edge matrix $B_K = [x_{i_1} - x_{i_0}, x_{i_2} - x_{i_0}, x_{i_3} - x_{i_0}]$. Assume $\det B_K > 0$. For $x \in K$, write

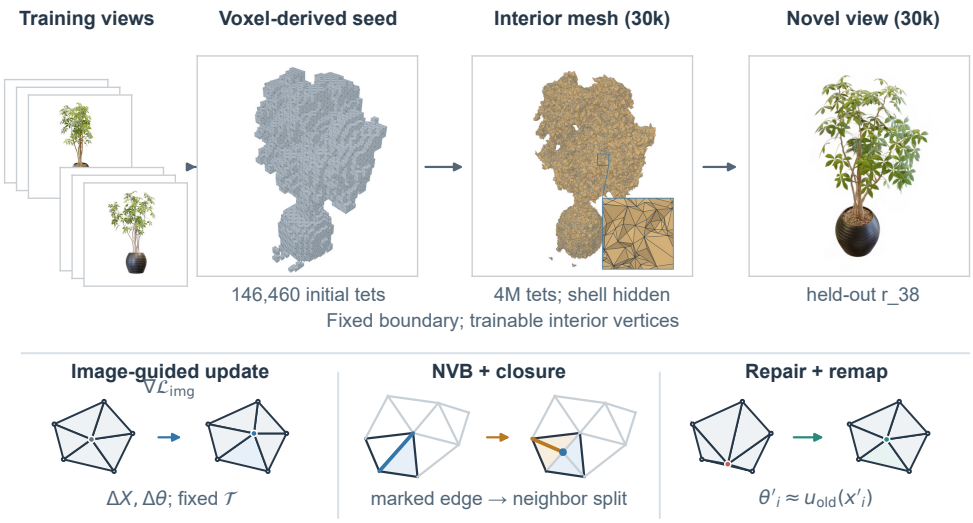

Figure 2: RevTet representation and operators on Ficus (NeRF Synthetic). Top: stacked training views, the voxel-derived seed, the exposed interior of the learned mesh, and a held-out rendering from the same camera. The anchored exterior is hidden to reveal internal tetrahedral faces; the inset magnifies the same mesh. Bottom: irregular 2D triangular analogues of vertex learning, NVB closure, and repair with optional remapping.

$\boldsymbol{\lambda}_K = (\lambda_1, \lambda_2, \lambda_3)^\top$; the weights and coefficient field are

$$\begin{aligned} \boldsymbol{\lambda}_K(x) &= B_K^{-1}(x - x_{i_0}), \qquad \lambda_0(x) = 1 - \mathbf{1}^\top \boldsymbol{\lambda}_K(x), \\ u_h(x) &= \sum_{j=0}^{3} \lambda_j(x)\theta_{i_j} = \sum_i \phi_i(x; X, \mathcal{T})\theta_i. \end{aligned} \tag{2}$$

Within the mesh domain, the global basis ϕ_i is piecewise affine and continuous across matching shared faces; density is defined to be zero outside this domain. Thus changing X changes the interpolation basis as well as the ray intervals, even if θ stays fixed. The density channel stores a pre-activation a_i, with $\sigma_h(x) = A(\sum_i \phi_i(x)a_i)$ for a nonnegative activation A. Appearance uses directional spherical harmonics (SH), with shared nodal coefficients and cell color offsets in all reported experiments; per-cell SH is an alternative configuration (Appendix B). Field-preservation statements below concern the shared affine coefficient channels.

A trained voxel field initializes X, $\mathcal{T}$, and the attributes. Rays traverse shared faces to obtain ordered cell intervals; midpoint density and constant or affine chord color approximate Eq. (1). The renderer $\mathcal{R}_r[X, \theta; \mathcal{T}]$ is differentiable almost everywhere on $\mathcal{G}(\mathcal{T})$ and supplies gradients to attributes and positions. Appendix B details the appearance variants and numerical checks.

3.2 Separating field learning from mesh maintenance

Vertex positions determine both where the radiance field is sampled and how its values are interpolated. We therefore distinguish two reasons for moving a vertex. During learning, positions and attributes change together to reduce image error. During fixed-connectivity geometric maintenance, positions change to improve cell geometry while disturbing the learned field as little as possible. Keeping connectivity fixed during these pure displacement steps preserves vertex identities and attribute correspondence; geometric checks restrict admissible motion. Fixed connectivity alone does not preserve the field: moving a vertex changes its interpolation weights even when stored attributes are unchanged. In-place repair reuses the recorded hierarchy to choose geometric corrections without changing active identities. Refinement and merging are separate topology events, each with its own transfer rule.

Learning changes the scene to fit images. On the current connectivity $\mathcal{T}$, we jointly optimize positions X and attributes θ:

$$\min_{X \in \mathcal{G}(\mathcal{T}), \theta} \quad \mathbb{E}_r \, \ell(\mathcal{R}_r[X, \theta; \mathcal{T}], I_r) + \sum_q \lambda_q \mathcal{L}_q. \tag{3}$$

Here I_r is the observed ray color, ℓ the image discrepancy, and $\mathcal{L}_q$ auxiliary regularizers. The feasible set $\mathcal{G}(\mathcal{T})$ fixes boundary vertices and requires positive cell volumes with a safety margin. Appendix B specifies these constraints and the correction/rollback checks. Connectivity changes occur only through the separate operators below.

Maintenance changes the mesh carrying the scene. To see why it needs an attribute-transfer rule, follow a point x during a valid fixed-connectivity move. Let $\Delta x_h(x)$ and $\Delta\theta_h(x)$ interpolate the vertex displacements and coefficient increments with the old barycentric weights. The moved point keeps those weights, so

$$u_{\text{new}}(x + \Delta x_h(x)) = u_{\text{old}}(x) + \Delta\theta_h(x). \tag{4}$$

With zero attribute increment, this equation preserves the value at the *moved* point, not at the original world-space position. For example, moving the vertices supporting a density transition moves that transition even if all stored values stay unchanged. A repair that aims to preserve the old scene should instead assign each relocated vertex the old field value at its *new* position. The derivation and continuous ALE interpretation are in Appendix A; the identity concerns interpolated coefficients, not physical conservation.

Consequences for adaptation. Each edit pairs geometry with field transfer: refinement restricts the old affine field, merging discards fine detail, and fixed-connectivity repair may remap approximately. Structural reversibility identifies undoable topology; transfer determines information loss.

3.3 Conforming reversible adaptation and mesh maintenance

Each adaptation operator combines a mesh edit, an attribute-transfer rule, and a validity check.

Refinement: from local LE to compatible NVB. Local longest-edge (LE) bisection targets long edges, but our independent-cell LE schedule does not synchronize shared faces and can leave hanging midpoints. We use newest-vertex bisection (NVB) with compatible edge tags, inherited vertex orderings, and recorded parent–child relations (Maubach, 1995; Stevenson, 2008). An image-sensitive split request induces a closure $\mathcal{C}(e)$ of neighboring splits that match shared faces. The complete event is accepted only if $|\mathcal{T}| + \Delta N(\mathcal{C}(e)) \le N_{\max}$, including closure cost in the active-cell budget. The comparison concerns our local LE schedule; LE methods can also achieve conformity with additional closure rules. All incident cells share one midpoint on the split edge (a, b):

$$x_m = \tfrac{1}{2}(x_a + x_b), \qquad \theta_m = \tfrac{1}{2}(\theta_a + \theta_b), \qquad u_{h'}|_{K_j} = u_h|_{K_j} \quad (j = 1, 2). \tag{5}$$

Each child K_j reproduces the parent's affine coefficient field in exact arithmetic (Proposition 1); cell-local appearance parameters are copied to both children. Numerical renders can still change because splitting ray intervals changes midpoint quadrature after nonlinear density activation. Figure 3 contrasts complete NVB closure with the independent local LE schedule.

Merging. An inverse event restores recorded parents only when their children are active leaves, every pair incident to the removed midpoint participates, and no external active cell still uses that midpoint. Restored cells must pass face-incidence, boundary, and determinant checks. Surviving nodal values are retained and child-local attributes are averaged. Before geometric deformation, a midpoint detail $d_m = \theta_m - \frac{1}{2}(\theta_a + \theta_b)$ measures information unavailable to the coarse affine interpolant; nonzero d_m makes merging lossy. Deformation and cell-local attributes introduce additional error. Low-density merging can release training capacity; the reported full-training experiments disable merging and evaluate inverse events through offline LOD.

Repair and remapping. For cells approaching degeneration, we replay recorded midpoint positions in ancestor-to-descendant order and apply bounded vertex-star corrections to residual risky cells. The evaluated in-place replay keeps connectivity, cell and vertex identities, birth-reference determinants, and stored field attributes unchanged; only geometry and the corresponding birth-position anchors are updated. Candidate corrections must reduce the configured risk energy and pass affected-cell and global geometric checks; failed updates are rolled back. Optional remapping approximates the old field at relocated vertices (Appendix B.2); it is tested separately and disabled in the scene ablations.

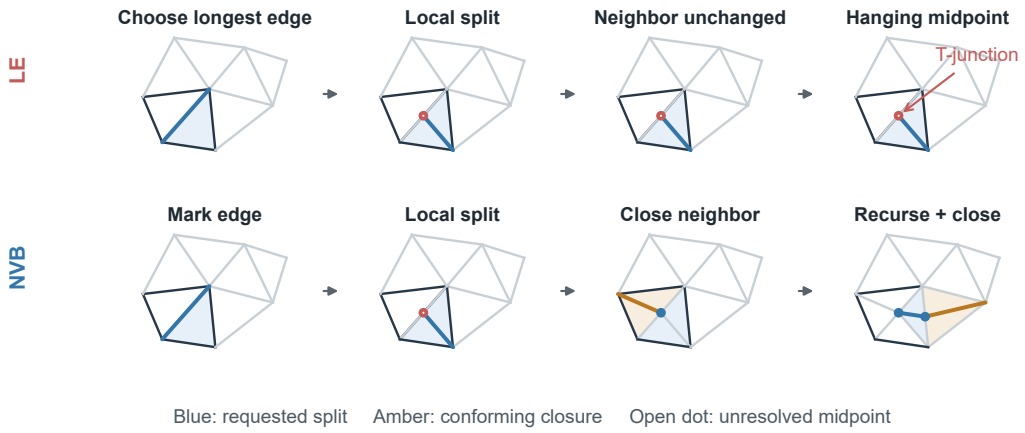

Figure 3: Planar analogy of local LE (top) and conforming NVB (bottom). Independent LE leaves a hanging midpoint; NVB closes the temporary mismatch, including recursively split children. Blue marks requested splits, amber marks closure, and open dots mark unresolved midpoints. The evaluated models use tetrahedra.

4 Experiments

We evaluate reconstruction on a conforming hierarchy, repair at comparable image fidelity, and representation-size reduction without retraining.

4.1 Evaluation protocol

NeRF Synthetic is primary: reconstruction and LOD cover eight scenes and repair four pairs, each with 200 test views at 800×800 (Mildenhall et al., 2020). We report equal-scene PSNR/SSIM and VGG LPIPS with $[-1, 1]$ inputs. LOD and repair use PNG renders; test views never guide adaptation. Unless noted, strict NVB uses seed 3721, 30k post-initialization iterations, and a 4M-tetrahedron ceiling. Certificates check lineage, orientation, face incidence, and boundary anchors; traversal counters cover dropped entries, folded exits, walk deaths, and iteration limits.

Matched operator ablations are separated from saved-system comparisons with different recipes. The Lego continuation and four-scene repair study each use one seed, so differences are descriptive, not significance estimates. Published results retain source protocols; visualization-only models are listed in Appendix B.3. Single-run timing and memory use one RTX 4090 (24 GB); time excludes initialization and checkpoint I/O, and memory is the CUDA allocator peak.

4.2 Reconstruction quality and adaptive refinement

We first place RevTet's reconstruction quality alongside established explicit representations. Table 1 separates published NeRF Synthetic results for 3DGS and SVRaster from our local evaluation of SVRaster and strict-NVB RevTet. The local rows use all 200 test views of all eight scenes. RevTet uses 30k tetrahedral-training iterations after voxel initialization and a 4M-cell ceiling; the local SVRaster checkpoints use 20k iterations. This is a reconstruction-quality reference, not a comparison at equal training cost or primitive count.

RevTet reaches approximately 33 dB mean PSNR, 0.26 dB below local SVRaster, with similar SSIM and higher LPIPS. These results establish a useful reconstruction baseline for studying conforming adaptation and subsequent mesh maintenance. All eight RevTet final states pass the recorded geometry checks, and strict traversal over the test views reports zero error counters. Our earlier LE variant attains higher mean PSNR under a different saved recipe (33.30 dB; Appendix D.3); neither comparison isolates the effect of conformity alone.

Table 1: NeRF Synthetic, equal-scene means over eight scenes. Published PSNR/SSIM are averaged from the per-scene values reported by Sun et al. (2025) for both methods; local LPIPS uses VGG with $[-1, 1]$ inputs. A dash denotes an unverified matching LPIPS convention, not an unavailable method. Source values and evaluation details are in Appendix D.2.

Method	Evaluation	PSNR↑	SSIM↑	LPIPS↓
3DGS (Kerbl et al., 2023)	Published	33.33	.9694	—
SVRaster (Sun et al., 2025)	Published	33.21	.9643	—
SVRaster	Local	33.25	.9643	.0387
RevTet (strict NVB)	Local	32.99	.9644	.0441

Why adapt both resolution and geometry? A matched Lego continuation isolates these choices at 500k and 800k cell ceilings. Starting from the same coarse checkpoint and using the same 6,000 updates, importance-guided refinement improves clamp-float PSNR over random allocation by 0.60 and 1.40 dB; vertex motion adds a further 0.68–0.69 dB. SSIM and LPIPS improve in the same direction, and all six states pass geometry and traversal checks (Appendix D.4). Together, these results show that the conforming representation supports learning across scenes and that its allocation and motion operators improve the tested capacity-limited continuation. The latter is a single-scene result, not a multi-scene scaling law.

4.3 Improving mesh quality with repair

Can repair improve the mesh beyond what image optimization alone provides? We compare repair on and off on Chair, Drums, Ficus, and Lego, using identical per-scene voxel initialization, non-repair settings, and 30k-step training budgets. Both arms disable merging and finish with 4M tetrahedra. Repair combines in-place hierarchy replay with bounded vertex-star correction, using relative-volume and shape thresholds of 0.10 and 0.33 and remaining active through the final training interval. Stored attributes are retained without optional remapping. Appendix C.1 gives the full protocol and an earlier repair setting.

Better worst-cell geometry and fewer strongly compressed cells. We measure shape using mean-ratio quality Q_K (one for a regular tetrahedron) and compression using relative volume $r_K = \det B_K / \det B_K^{\text{birth}}$. Table 2 shows that the mean scene minimum $Q_{\min}$ rises by 23.4%, while the mean minimum relative volume $D_{\min}$ rises from 0.00160 to 0.06167. The fraction with $r_K < 0.05$ falls by 60.1%, so the gain extends beyond a single extreme cell. On Chair, Drums, and Lego, this strongly compressed population disappears, and $Q_{\min}$ improves by 20.4–37.7%. Ficus remains difficult: its minima improve only slightly in absolute terms, and the fraction below $r_K = 0.10$ increases from 1.024% to 1.034%.

Table 2: Four-scene repair ablation. Equal-scene means, one seed, 30k iterations, and 200 test views per scene. Minima are averaged over scenes. Both arms use identical diagnostic thresholds. All changes use unrounded values; image-metric changes are absolute, so displayed means can differ by rounding.

Metric	No repair	Repair	Change
$Q_{\min}$ ↑	0.17202	**0.21234**	**+23.4%**
$D_{\min}$ ↑	0.00160	**0.06167**	38.5× the baseline
Cells with $r_K < 0.05$ ↓	0.12911%	**0.05147%**	−60.1%
Cells with $r_K < 0.10$ ↓	0.68378%	**0.29000%**	−57.6%
PSNR↑ (dB)	32.8350	32.8289	−0.0062
SSIM↑	0.97425	0.97420	−0.00005
LPIPS↓	0.03063	0.03074	+0.00011
Training loop (s)	1017.7	1231.2	+21.0%
Peak allocated memory (GiB)	5.15	5.87	+14.2%

Local geometry makes the effect visible. Figure 4 compares two Lego regions in the final trained meshes. The highlighted cells in the repaired mesh have less compressed volume and higher shape quality, while patch-minimum relative volumes rise from 0.0029 to 0.0863 and from 0.0023 to 0.1002.

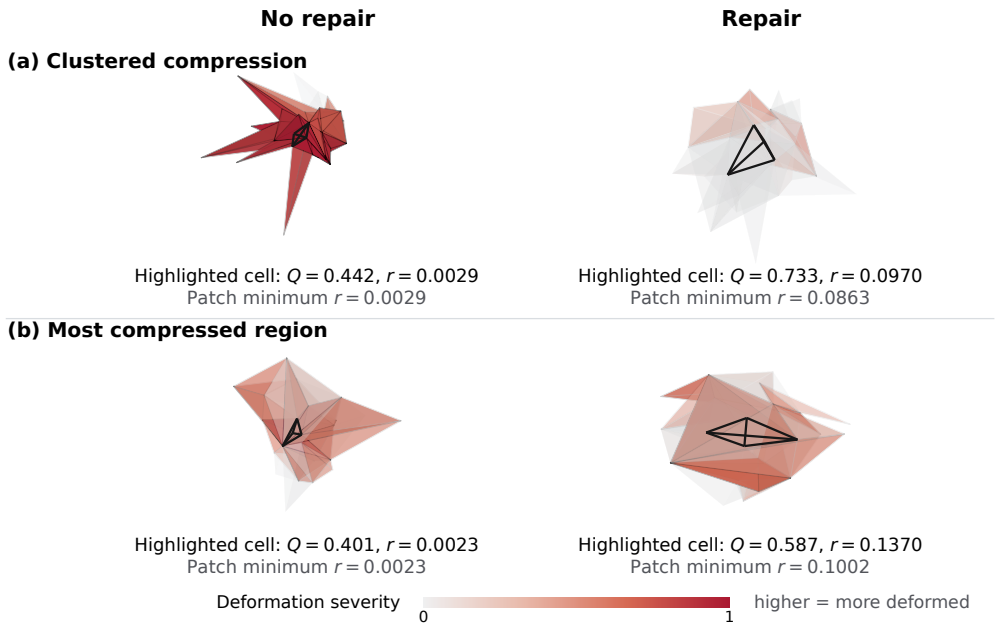

Figure 4: Repair alleviates local compression on Lego. Each row compares 56-cell patches in the same world-space neighborhood of two separately trained meshes, with a shared camera, spatial scale, and color mapping. Redder, more opaque cells have higher combined shape/volume deformation severity; black outlines identify the cells whose Q and r are reported. The examples are selected from a large control-arm risk cluster (a) and the cluster containing its minimum-r cell (b). They are spatial comparisons, not cellwise before/after pairs; selection and encoding are specified in Appendix C.1.

These examples illustrate localized improvement; they do not imply that every neighboring cell improves. Across the four scenes, Q_5 is almost unchanged and Q_1 decreases slightly (Appendix C.1), consistent with repair targeting severe defects rather than uniformly regularizing the mesh.

The mesh improves at a small measured image cost. Mean PSNR changes by -0.0062 dB, with every scene within ± 0.030 dB; SSIM and normalized VGG LPIPS change by -0.00005 and $+0.00011$. All eight runs pass the final geometry certificate, have no nonpositive cells, and record zero traversal failures over 200 views. Repair therefore strengthens geometric quality at comparable image fidelity, with a 21.0% increase in training-loop time. The result supports repair as a complement to image-driven learning: it improves the most vulnerable spatial support without requiring a substantial change in the learned appearance.

4.4 Image quality at reduced resolution

We next ask how much of a learned radiance field can be retained when its active mesh is coarsened without retraining. Starting from each of the eight strict-NVB checkpoints in Section 4.2, we compare importance-ranked legal inverse events (Importance) with seeded random event selection (RandomEvent) at six budgets from 100% to 50%. Importance uses only 32 half-scale training views. Both selectors start from the same full model; the tiers form a branching family rather than one nested path. A separate SVRaster comparison applies native maximum-weight voxel pruning to eight 20k checkpoints. All results use the same 200 PNG test views per scene; Appendix D.5 gives the branching protocol and exact retained counts.

Importance preserves more quality across the tested budgets. At approximately 80% retained cells, Importance loses only 0.16 dB mean PSNR relative to the full model, compared with 0.36 dB for RandomEvent; at 50%, the losses are 1.52 and 1.80 dB. Its mean advantage spans 0.11–0.33 dB across the five coarsened tiers, with lower LPIPS at every tier as well. All 88 RevTet states pass geometry certificates and traversal checks. The effect varies by scene, and indivisible events introduce small count differences: the maximum difference between the selectors' mean retained

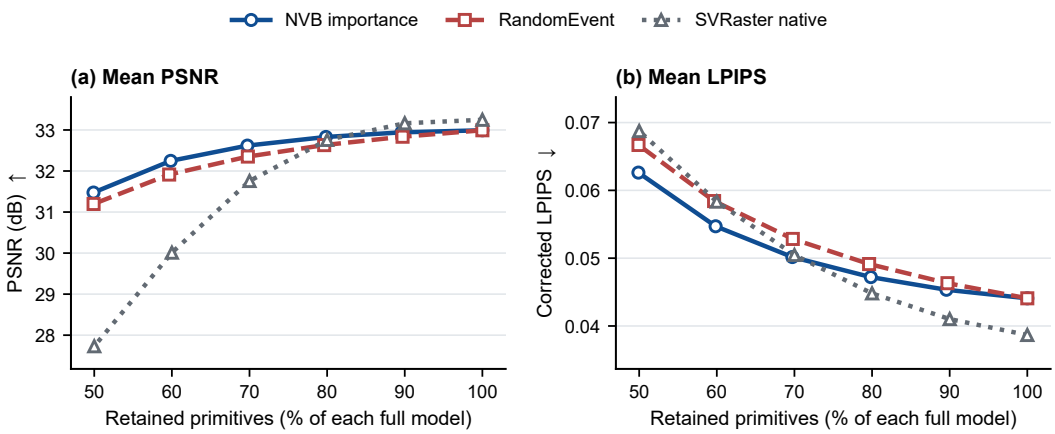

Figure 5: Image quality versus retained primitive fraction, averaged over eight scenes with no recovery training. Importance preserves higher mean PSNR and lower LPIPS than RandomEvent at all five coarsened budgets. Fractions are relative to each method's own full model: RevTet counts tetrahedra and SVRaster counts voxels, so equal percentages do not imply equal capacity, storage, or training cost.

fractions is 0.224 percentage points. The results therefore support the measured quality-retention benefit of importance ranking without claiming exact equal-count attribution.

Coarsening yields a useful, but lossy, size–quality trade-off. SVRaster starts with higher mean PSNR (33.25 versus 32.99 dB) and remains higher at 90%; at 80%, Importance is slightly higher (32.83 versus 32.76 dB). At 50%, the respective means are 31.48 and 27.73 dB. These curves describe sensitivity to each system's own post-training reduction, not superiority at matched absolute capacity. For RevTet, retaining about half the active cells reduces the persistent tensor inventory from 852.41 to 615.00 MiB, because the hierarchy itself remains substantial. Lower-budget endpoints on Chair and Drums indicate that greedy inverse events can run out; they do not establish an eight-scene curve below 50% (Appendix D.6). The practical outcome is adjustable resolution with explicit geometric checks and a measured image-quality cost.

5 Discussion and Limitations

Complementary roles. One persistent hierarchy supports allocation by refinement and motion, targeted geometric repair, and post-training resolution reduction, with explicit transfer rules for each edit. Evidence covers bounded synthetic scenes; allocation and motion use a controlled single-scene continuation.

Costs and limits. Repair targets worst cells rather than uniformly improving shape: Ficus retains defects, percentiles barely change, and measured training time rises 21%. Rendering speedups and fewer traversal failures remain unverified. Connectivity is reversible, not appearance: coarsening loses detail and fixed repair attributes alter the world-space interpolant.

Scope and scalability. External references use unmatched recipes, and per-method LOD fractions do not equalize storage or compute. History and closure consume memory and capacity; inverse events can exhaust. Evidence is single-seed, with four repair scenes and one continuation scene; broader claims require independent repair-policy validation and matched unbounded-scene comparisons.

6 Conclusion

RevTet unifies direct radiance-field learning, conforming refinement, repair, and validity-checked coarsening in a persistent tetrahedral hierarchy. Controlled continuations, four-scene repair ablations, and eight-scene coarsening show complementary benefits in capacity-limited reconstruction, worst-cell geometry, and post-training resolution control, while exposing trade-offs among geometry, image fidelity, and size.

Reproducibility statement

The appendix contains essential derivations, implementation details, and protocols for the reported supplementary checks and LOD curve. The accompanying experimental records specify code versions, configurations, commands, and source measurements for the reported results.

AI use statement

Generative AI tools, including OpenAI Codex, assisted with literature retrieval; method implementation and auditing; formulation and experiment-design feedback; diagnostic scripts and data-derived figures; result interpretation; and manuscript drafting, translation, editing, and organization. Reported measurements were produced by the recorded experimental pipelines, not generated by AI. AI-assisted material was checked against cited sources, executable tests, and recorded measurements where applicable. The authors reviewed the final text, claims, code, figures, and experimental provenance and take responsibility for the complete submission.

References

Jean Donea, Antonio Huerta, Jean-Philippe Ponthot, and Antonio Rodríguez-Ferran. Arbitrary lagrangian-eulerian methods. In Erwin Stein, René de Borst, and Thomas J. R. Hughes (eds.), *Encyclopedia of Computational Mechanics*, volume 1, chapter 14, pp. 413–437. John Wiley & Sons, 2004. doi: 10.1002/0470091355.ecm009. URL https://onlinelibrary.wiley.com/doi/10.1002/0470091355.ecm009.

Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant Foam: Real-time differentiable ray tracing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4135–4145, 2025. doi: 10.1109/ICCV51701.2025.00394. URL https://arxiv.org/abs/2502.01157.

C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3):227–253, 1974. doi: 10.1016/0021-9991(74)90051-5.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. doi: 10.1145/3592433. URL https://arxiv.org/abs/2308.04079.

Yaron Lipman. Bijective mappings of meshes with boundary and the degree in mesh processing. *SIAM Journal on Imaging Sciences*, 7(2):1263–1283, 2014. doi: 10.1137/130939754. URL https://arxiv.org/abs/1310.0955.

Alexander Mai, Trevor Hedstrom, George Kopanas, Janne Kontkanen, Falko Kuester, and Jonathan T. Barron. Radiance Meshes for volumetric reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8267–8277, 2026. URL https://arxiv.org/abs/2512.04076.

J. M. L. Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *SIAM Journal on Scientific Computing*, 16(1):210–227, 1995. doi: 10.1137/0916014.

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 *Computer Vision – ECCV 2020*, volume 12346 of *Lecture Notes in Computer Science*, pp. 405–421. Springer, 2020. doi: 10.1007/978-3-030-58452-8_24. URL https://arxiv.org/abs/2003.08934.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127. URL https://arxiv.org/abs/2201.05989.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Rob Stevenson. The completion of locally refined simplicial partitions created by bisection. *Mathematics of Computation*, 77(261):227–241, 2008. doi: 10.1090/S0025-5718-07-01959-X.

Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. Sparse voxels rasterization: Real-time high-fidelity radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16187–16196, 2025. URL https://arxiv.org/abs/2412.04459.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. Learning controllable adaptive simulation for multi-resolution physics. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PhktEpJHU3.

A Essential derivations

A.1 Learning objective and field update

Equation (3) combines image supervision with auxiliary regularization. Here we detail the moving-basis relation underlying Eq. (4).

For a valid fixed-connectivity move, interpolate nodal increments as $\Delta x_h(x) = \sum_i \phi_i(x)\Delta x_i$ and $\Delta\theta_h(x) = \sum_i \phi_i(x)\Delta\theta_i$, using the old barycentric weights. The piecewise-affine map, assumed injective and nondegenerate on each cell, preserves these weights at the moved point, so

$$u_{\text{new}}(x + \Delta x_h(x)) = \sum_i \phi_i(x)(\theta_i + \Delta\theta_i) = u_{\text{old}}(x) + \Delta\theta_h(x). \tag{6}$$

This proves Eq. (4) componentwise. In continuous optimization time τ, define mesh velocity $w_h = \sum_i \dot{x}_i\phi_i$ and carried coefficient rate $s_h = \sum_i \dot{\theta}_i\phi_i$. Differentiation inside each cell gives the ALE form (Hirt et al., 1974; Donea et al., 2004)

$$\partial_\tau u_h + w_h \cdot \nabla u_h = s_h. \tag{7}$$

This is a moving-basis identity, not a physical conservation law. Holding coefficients fixed sets $s_h = 0$, whereas keeping the world-space field fixed requires $s_h = w_h \cdot \nabla u_h$. Optional repair remapping approximates the latter condition through nodal resampling; exact preservation is unavailable when the old field is not representable on the repaired mesh.

A.2 Refinement preservation and coarsening loss

Proposition 1 (Nested refinement)**.** *For an exact midpoint bisection of a nondegenerate tetrahedron, set $x_m = (x_a + x_b)/2$ and $\theta_m = (\theta_a + \theta_b)/2$. Restricting the parent's affine field to its children preserves that field pointwise. For a conforming sequence of such bisections, the global shared nodal pre-activation field is unchanged in exact arithmetic.*

An affine field is uniquely determined by its vertex values. The averaged endpoint value equals the parent field at the new midpoint, so each child interpolates the same parent field. Compatible shared-face subdivisions extend this argument across the mesh. Conversely, coarsening discards any midpoint detail learned after insertion. For unchanged nested geometry, coarsening is lossless in the shared coefficient channels exactly when the fine field lies in the coarse space. Geometric deformation and cell-local appearance require additional compatibility conditions.

Coefficient preservation does not imply identical numerical renders. Density is activated after interpolation and evaluated at chord midpoints; subdividing a chord changes this quadrature even if the underlying coefficient field is unchanged.

B Implementation details

B.1 Rendering and geometric checks

The density channel stores a pre-activation a_i. The renderer uses $\sigma_h(x) = A(\sum_i a_i\phi_i(x))$, where A is the configured nonnegative activation. The base synthetic mode combines shared nodal SH with cell color offsets; per-cell SH is also supported. Affine-field preservation applies to the interpolated coefficient channels, not all derived appearance computations. Rendering traverses shared faces and uses midpoint density with constant or affine chord color.

For tetrahedron K, let $D_K(X) = \det B_K(X)$ be six times its signed volume and $D_K^\star > 0$ the stored reference. Boundary vertices $i \in B$ remain at anchors a_i^{bdry}, including the birth positions of new boundary midpoints. The fixed-connectivity feasible set in Eq. (3) is

$$\mathcal{G}(\mathcal{T}) = \left\{ X : x_i = a_i^{\text{bdry}} \ (i \in B), \quad D_K(X) \geq \max(\epsilon D_K^\star, \delta) \ (K \in \mathcal{T}) \right\}. \tag{8}$$

Here $\epsilon > 0$ is the relative safety threshold and $\delta = 10^{-12}$ an absolute determinant floor. The reference is halved for midpoint children and summed for merged siblings. Topology edits are checked for

positive determinants, face incidence, and boundary preservation; inverse events additionally check their local fans and neighboring rings. After repeated refinement, $\epsilon D_K^\star$ can fall below δ; the binding floor then guards against numerical degeneracy rather than enforcing a scale-relative margin.

Global non-overlap requires more than positive cell volumes: the initial mesh must be a valid oriented simplicial manifold, the deformation continuous and orientation-preserving, and the boundary map bijective (Lipman, 2014). Our floating-point checks support these conditions but are not exhaustive collision detection. Refinement uses rendering sensitivity for priority; training-time merging selects complete events whose corner densities are below a threshold. Closure cost counts toward the active-cell budget, which does not bound history or optimizer memory.

B.2 Repair and attribute transfer

Repair is triggered by relative-volume and shape-quality thresholds. It replays midpoint positions in hierarchy order, then uses bounded local vertex adjustments for remaining risky cells. All affected cells are checked, and unsuccessful updates are rolled back. Thresholds must accommodate useful anisotropy rather than treating every thin cell as invalid. The main repair experiments use in-place replay: midpoint positions are recomputed through the recorded hierarchy without reallocating cell or vertex IDs or changing stored field attributes and birth references. This differs from the earlier rebuild-based replay, which collapsed and re-expanded events and could reindex a large part of the active mesh. Results from that older implementation are not used as evidence for the main repair ablation. The full-training protocol, fixed-threshold diagnostics, and local geometry comparisons are detailed in Appendix C.1.

For a pure position correction $x_i' = x_i + \delta x_i$, the optional remapping rule estimates the old field at the new position:

$$\begin{aligned} \theta_i' &\simeq u_h^{\text{old}}(x_i') \simeq \theta_i + G_i^\top \delta x_i, \\ G_i &= \arg\min_G \left[\sum_{j \in \mathcal{N}_i} w_{ij} \|G^\top e_{ij} - (\theta_j - \theta_i)\|^2 + \lambda_i \|G\|_F^2 \right]. \end{aligned} \tag{9}$$

Here $e_{ij} = x_j - x_i$, $w_{ij} = \|e_{ij}\|^{-2}$, and $G_i \in \mathbb{R}^{3\times m}$ for m shared nodal channels; each column estimates one coefficient's spatial gradient. Ridge regularization and bounded corrections stabilize the fit; gradient discontinuities and clipping make transfer approximate. Remapping is evaluated in a separate field-transfer test; the scene repair ablations use geometric correction without this optional step.

The gradient fit in Eq. (9) has the regularized normal equations

$$\left(\sum_{j \in \mathcal{N}_i} w_{ij} e_{ij} e_{ij}^\top + \lambda_i I_3 \right) G_i = \sum_{j \in \mathcal{N}_i} w_{ij} e_{ij} (\theta_j - \theta_i)^\top. \tag{10}$$

The ridge weight λ_i scales with the trace of the local edge matrix, and $\|\cdot\|_F$ denotes the Frobenius norm. Corrections are clipped relative to neighboring attribute differences; incident edges retain their tetrahedral multiplicities. Regularization, clipping, and crossing coefficient-gradient jumps make the transfer approximate. Position moments of moved vertices and attribute moments of remapped parameters are cleared; topology edits rebuild optimizer state according to the trainer policy.

B.3 Mesh visualizations

Figure 1 shows the held-out r38 view of Lego and Hotdog. Within each scene, the reference and LOD renders use the same crop and scale. Local cells are selected from checkpoint geometry in a fixed spatial neighborhood; an upper-front window exposes their interior. Directional shading follows face normals, and an 8% centroid-based display shrink separates facets without modifying the represented mesh. Ochre is assigned to faces near the window, independently of density, refinement level, or cell quality. Lego retains 25% at the lowest displayed tier; The visualization-only Hotdog state retains about 35% when eligible inverse events are exhausted. These visualization models are separate from the quantitative LOD checkpoints in Fig. 5 and Appendix D.

Figure 2 uses the Ficus strict-NVB 30k checkpoint trained on NeRF Synthetic. Its 146,460-cell initialization is recovered from the persistent root cells at their initialization coordinates; it includes two uniform bisection rounds after converting each retained voxel to five tetrahedra. The learned mesh has 4,000,000 tetrahedra and 741,795 vertices. The seed, interior mesh, and held-out r_38 rendering use the same camera, perspective projection, and image crop. For visibility only, the anchored exterior and tetrahedra incident on boundary vertices are hidden; the exposed faces of the remaining interior tetrahedra are rendered at their learned coordinates. No geometry is moved, shrunk, or retriangulated. The inset magnifies those same projected triangular faces. The lower panels and Fig. 3 are irregular planar triangular analogies, not tetrahedral slices or measured meshes.

C Supplementary validation and evaluation protocol

C.1 Repair ablation: protocol, per-scene results, and visual evidence

Matched training protocol. The repair comparison covers the four completed scene pairs: Chair, Drums, Ficus, and Lego. The unfinished Hotdog run is excluded. Each pair uses the same stage-1 checkpoint, code revision, seed 3721, non-repair configuration, and 30k training iterations, with merging disabled and a 4M-cell ceiling. The enabled policy (C in the experiment records) uses in-place replay and vertex-star correction at 1,000-step intervals from step 2,000 through 30,000. Its trigger thresholds are $r_K < 0.10$ or $Q_K < 0.33$, with caps of 1,024 replay events and 4,096 vertex-repair candidate cells, 16 vertex-optimization steps, and up to three vertex passes. These settings control the repair workload; acceptance checks still constrain each update. No optional attribute remapping is used.

Metric definitions and consistent thresholds. Mean-ratio quality is $Q_K = 6(\sqrt{2}|\det B_K|)^{2/3}/\sum_e \|e\|^2$, summed over the six edges; $Q_K = 1$ for a regular tetrahedron. Relative volume is $r_K = \det B_K / \det B_K^{\text{birth}}$. Q_p and D_p denote within-scene percentiles, then averaged equally over scenes; minima are also averaged by scene. Birth references are preserved during in-place repair. The raw audit's risk union uses different thresholds in the two arms and is not compared. We instead recompute fixed-threshold fractions for both arms. All eight final meshes have $Q_{\text{min}} > 0.08$, so the fixed union $r_K < 0.05$ or $Q_K < 0.08$ coincides with $r_K < 0.05$. Image metrics use 200 PNG test views; LPIPS uses the corrected VGG evaluation on $[-1, 1]$ inputs. Training-loop time excludes initialization and checkpoint I/O. The corresponding complete training-step wall times average 1026.6/1239.5 seconds (no repair/repair, +20.7%).

Table 3: Per-scene repair results. Entries are no repair / repair; ΔPSNR is repair minus no repair in dB. Fractions use fixed thresholds in both arms.

Scene	$Q_{\text{min}} \uparrow$	$D_{\text{min}} \uparrow$	$r_K < 0.05$ (%)	$r_K < 0.10$ (%)	ΔPSNR
Chair	.2115/.2599	.00159/.08636	.07815/.00000	.4894/.0214	−.0201
Drums	.2127/.2562	.00230/.07880	.09755/.00000	.6179/.0651	−.0126
Ficus	.0810/.0813	.00020/.00056	.21472/.20587	1.0240/1.0344	+.0292
Lego	.1829/.2519	.00231/.08095	.12603/.00000	.6038/.0391	−.0211

Where the gains occur. The final Q_{min} and D_{min} improve in all four pairs, but the effect is concentrated in Chair, Drums, and Lego (Table 3). Mean D_1 rises from 0.11733 to 0.12756, improving in three scenes; Ficus decreases from 0.09904 to 0.09864. The mean fraction with $r_K < 0.01$ falls from 0.006756% to 0.003100%, with all remaining cells below this threshold in Ficus. Shape percentiles behave differently: Q_1 changes from 0.34481 to 0.34418, and Q_5 from 0.35048 to 0.35050. Repair therefore improves the worst-cell shape floor and substantially reduces severe compression, but does not shift the entire quality distribution upward. The precise cause of the residual Ficus defects is not isolated by this experiment.

Comparison with the earlier repair policy. The earlier setting uses thresholds 0.10/0.15, stops repair at 24k, and caps replay events and vertex candidates at 512 each, with 12 vertex-optimization steps. Table 4 compares it with the current policy on exactly the same four scenes. The current policy yields a higher mean worst-cell shape quality and a much higher relative-volume floor. It changes both activation thresholds and repair workload, so the comparison evaluates the combined policy rather than isolating any one parameter. This setting was selected through exploratory probes,

and the four-scene, single-seed results should not be read as an independent test of generalization or multi-seed robustness.

Table 4: Repair-policy comparison on the same four scenes. All runs train for 30k steps; only the repair settings differ.

Metric	No repair	Earlier policy	Current policy
$Q_{\text{min}} \uparrow$	.17202	.17908	**.21234**
$D_{\text{min}} \uparrow$	.00160	.00306	**.06167**
$D_1 \uparrow$	.11733	.11885	**.12756**
$r_K < 0.10$ (%)$\downarrow$	.68378	.64783	**.29000**
PSNR$\uparrow$ (dB)	32.8350	32.8327	32.8289
Training loop (s)	1017.7	1097.2	1231.2

Figure selection and reproducibility. Figure 4 uses the final 30k Lego checkpoints from these paired runs. Control-arm cells satisfying $r_K < 0.05$ or $Q_K < 0.08$ are grouped by face connectivity. Panel (a) comes from a 107-cell risk component; panel (b) comes from the 24-cell component containing the global minimum-r_K cell. Each displayed patch contains 56 neighboring cells grown around the component's minimum-r_K cell. The repaired patch is seeded at the cell nearest the control patch center. The black outlines mark these two seed cells, which are spatially associated but do not share an identity across runs. Geometry is read directly from the checkpoints. Each row uses the joint bounding box of its two patches, an orthographic camera at elevation 18° and azimuth -58°, and a common scale. The repaired patches have lower local minimum Q (.3269 versus .3375; .3257 versus .3511), even though the highlighted cells have higher Q; the image illustrates reduced compression rather than a uniform local shape improvement.

Color and opacity encode a shared deformation severity $S = \max(s_Q, s_r)$, where $s_Q = \text{clip}_{[0,1]}[(0.46409 - Q)/(0.46409 - 0.18290)]$ and $s_r = \text{clip}_{[0,1]}[(0.10 - r)/(0.10 - 0.002313)]$. These anchors are fixed across all four panels, using the lower of the two scene-wide medians for the shape reference and the two-arm minima for the lower bounds. Color follows $S^{0.55}$ and opacity is $0.18 + 0.77S$; outlined cells have opacity at least 0.35. Thus the legend is a joint shape/volume severity scale, not a direct Q colorbar. The source package includes the selected coordinates, per-cell metrics, rendering script, and camera parameters.

C.2 Field-transfer check

A CPU float64 test uses seed 20260923 and random affine scalar fields on four-tetrahedron fans. The outer tetrahedron is randomly rotated and anisotropically scaled with log-scales from $[-0.55, 0.55]$ normalized to unit product, then normalized to mean center-to-corner distance $h = 1$. Only its center moves. Each scale requests 600 trials; moves that invert a cell or reduce its volume below 5% of its initial value are rejected. We evaluate 128 uniform barycentric samples per original cell (512 fixed world-space points per accepted trial) and normalize error by $\|\nabla u\| h$. The gradient fit uses inverse-squared edge weights, ridge $10^{-4}\,\text{tr}(\sum wee^\top)$, and a correction cap of 0.25 times the maximum incident attribute difference.

Table 5: Full-domain normalized field RMSE. Each row requests 600 independent fans. The oracle uses the known affine gradient.

$\|\delta x\|/h$	Accepted	Frozen values	Bounded remap	Oracle
.01	600	1.868×10^{-3}	8.504×10^{-7}	1.224×10^{-16}
.05	600	9.130×10^{-3}	4.436×10^{-6}	1.224×10^{-16}
.10	600	1.824×10^{-2}	8.619×10^{-6}	1.181×10^{-16}
.20	600	3.578×10^{-2}	1.590×10^{-3}	1.267×10^{-16}
.40	268	7.295×10^{-2}	2.351×10^{-2}	1.325×10^{-16}

Remapping reduces error for small displacements, but its error grows with large moves; at $0.4h$, only 268 of 600 proposals are accepted. These are acceptance-conditioned scalar-field tests, not scene-rendering improvements or a general accuracy guarantee.

C.3 Operator and rendering checks

Fifteen CUDA assertions cover merging, affine inheritance, geometric projection, and repair. On a trained Lego mesh with 390,617 cells and 87,872 vertices, one refinement request splits three parents through closure; a complete inverse restores counts, canonical connectivity, boundary anchors, and reference determinants. Intermediate audits report no nonpositive cells, nonmanifold faces, or folded faces. These are executed local checks, not an exhaustive guarantee.

An independent NumPy float64 reference intersects rays with all tetrahedra and sorts their intervals without using CUDA adjacency or entry buffers. Across 16,384 rays on four small meshes, maximum color and transmittance errors against the CUDA renderer are 1.383×10^{-6} and 1.571×10^{-6}, below the preselected 10^{-4} tolerance. The reference shares the renderer's quadrature, so it tests traversal and forward implementation, not exact physical integration. Large adaptive states still require broader stress testing.

A saturated-checkpoint control found less than 0.002 dB PSNR difference among its four fixed/movable and fixed/reallocated arms. This indicates little image-quality response in that saturated regime, in contrast to the coarse-state continuations in Appendix D.4.

D Supporting tables and plots

D.1 Standard volume-rendering equations

Equation (1) in Section 2 defines the continuous model. For computation, ordered ray intervals give the following discrete approximation.

Approximating density and color as constant over each of N ordered ray intervals, with length Δ_i and values σ_i, c_i, gives opacity $\alpha_i = 1 - \exp(-\sigma_i \Delta_i)$ and accumulated transmittance $T_i = \prod_{j<i}(1-\alpha_j)$. Front-to-back alpha compositing then yields

$$\widehat{C}(r) = \sum_{i=1}^{N} T_i \alpha_i c_i + T_{N+1} c_{\text{bg}}. \tag{11}$$

Here $T_1 = 1$ and $T_{N+1} = \prod_{j=1}^{N}(1-\alpha_j)$. This formulation connects local field values to image observations. Representations differ in how these values and their spatial support are parameterized and how rendering is approximated. In particular, 3DGS also uses alpha compositing, but its projected Gaussian opacities are not generally the exact interval integrals of Eq. (1).

Table 6: Structural comparison of the original methods. "Inverse hierarchy" means an explicit compatible parent–child split/coarsen construction, not general adaptivity. This table does not rank reconstruction quality or speed.

Method	Support and attributes	Adaptation and hierarchy	Relevant trade-off
NeRF	Continuous coordinate MLP; no explicit cell adjacency	Network optimization; no cell inverse hierarchy	Flexible implicit queries; original method needs neural evaluation and ray samples
3DGS	Overlapping anisotropic Gaussians; direct opacity and SH	Clone, split and prune; no conforming cell inverse hierarchy	Efficient splats; no explicit volume partition
SVRaster	Sparse voxels; corner density, cell SH	Octree-scale split/prune; flat leaves, no reported inverse hierarchy	Structured allocation; axis-aligned supports
Radiant Foam	Voronoi cells; learned cell attributes and adjacency	Optimize sites and tessellation; no reported inverse hierarchy	Deformable cells and ray tracing; non-simplicial supports
Radiance Meshes	Delaunay tets; hash-grid neural field supplies attributes	Move/add sites, retriangulate; no reported inverse hierarchy	Adaptive mesh output; neural queries accommodate connectivity changes
RevTet	Conforming tets; direct nodal and optional cell attributes	Constrained move, NVB split/inverse merge, repair	Explicit transfer lineage; closure cost and fixed boundary

D.2 External reconstruction references and metric conventions

The published rows in Table 1 come from Table 22 on page 20 of the SVRaster project paper and supplement (Sun et al., 2025), retrieved on 26 September 2026. That table reports both 3DGS and SVRaster; the 3DGS row here is therefore the comparison reported by Sun et al., rather than a separate reproduction or a value copied from the original 3DGS paper. Table 7 reproduces the relevant PSNR/SSIM values. The main-table means are computed from these rounded scene values, which limits their precision. Published and local results retain separate provenance; matching the dataset name alone does not establish identical preprocessing or evaluation implementations.

Table 7: Published NeRF Synthetic values reported in SVRaster's Table 22. Means are arithmetic averages of the eight printed rows.

Scene	PSNR↑		SSIM↑	
	3DGS	SVRaster	3DGS	SVRaster
Chair	35.89	35.91	.987	.986
Drums	26.16	26.09	.955	.947
Ficus	34.85	34.37	.987	.984
Hotdog	37.67	37.42	.985	.984
Lego	35.77	35.54	.983	.981
Materials	30.01	30.00	.960	.954
Mic	35.38	36.00	.991	.992
Ship	30.92	30.38	.907	.886
Mean	33.33	33.21	.9694	.9643

The official SVRaster repository distinguishes its default LPIPS evaluation on $[0, 1]$ RGB from corrected evaluation on $[-1, 1]$ RGB. We do not assume that the published LPIPS entries use the corrected convention, and omit them from the main comparison. Both local rows use corrected VGG LPIPS on the same 200 PNG test views at 800×800. The local SVRaster values are the unpruned full-model states used in Section 4.4; their training provenance is described in Appendix D.5. No new 3DGS training or equal-compute comparison is implied.

D.3 Saved strict-NVB and LE variants

Table 8 retains the comparison with our earlier local longest-edge (LE) system. Both variants use the eight scenes and 200 test views with corrected VGG LPIPS, but their saved training configurations and rendering backends are not fully matched. These results describe available systems rather than isolating the subdivision rule.

Table 8: NeRF Synthetic, 200 test views per scene. Higher PSNR/SSIM and lower corrected LPIPS are better. The last row is the equal-scene arithmetic mean; strict and LE are distinct saved system variants.

Scene	PSNR↑		Δ	SSIM↑		LPIPS↓	
	NVB	LE	dB	NVB	LE	NVB	LE
Hotdog	37.536	37.916	-.380	.98414	.98553	.02922	.02518
Chair	35.589	35.523	+.067	.98626	.98675	.01998	.01717
Drums	25.842	26.016	-.174	.94427	.94688	.06229	.05713
Ficus	34.701	34.845	-.144	.98543	.98600	.01843	.01585
Lego	35.185	35.738	-.554	.98085	.98294	.02197	.01993
Materials	29.866	30.056	-.190	.95496	.95682	.05047	.04302
Mic	35.128	35.194	-.067	.99076	.99104	.01160	.01134
Ship	30.096	31.130	-1.033	.88846	.89754	.13870	.11833
Mean	32.993	33.302	-.309	.96439	.96669	.04408	.03849

Strict NVB attains 32.993 dB mean PSNR versus 33.302 dB for LE, a -0.309 dB difference under these saved recipes; the LPIPS mean is .04408 versus .03849. The result demonstrates that strict conforming training runs to completion across all eight bounded scenes, while the available LE system

remains slightly stronger on mean image quality. It does not establish that the quality gap is caused by conformity or by the subdivision rule.

D.4 Capacity, geometry mobility, and refinement selection

We run a controlled short continuation from one strict-NVB Lego checkpoint (390,617 tetrahedra at iteration 1,000). Each arm receives the same 6,000 updates and the same refinement schedule, loss, seed, and optimizer settings; the final checkpoint is at iteration 7,000. We cross 500k/800k tetrahedron ceilings with frozen or trainable vertices under render-gradient importance, and include frozen-geometry random-priority controls at both ceilings. The movable arms enable vertex optimization after the shared checkpoint. We evaluate all six checkpoints on the same 200 test views, with clamp-float PSNR/SSIM and corrected VGG-LPIPS. No test views are used for training or selection.

Allocation and motion both improve the measured continuation. At 500k cells, random/frozen, importance/frozen, and importance/movable achieve 27.950, 28.549, and 29.234 dB, respectively; at 800k they achieve 28.565, 29.960, and 30.636 dB. Thus importance adds 0.599/1.395 dB over random allocation, and motion adds a further 0.685/0.676 dB over frozen importance. SSIM and LPIPS improve in the same direction. All six states finish within three cells of their ceilings and pass the recorded geometry and strict-walk checks. Figure 6 in Appendix D reports all three metrics against payload (about 96 MiB at 500k and 161 MiB at 800k).

These comparisons support importance allocation and movable vertices under this continuation schedule. Since moving vertices can also change later refinement priorities, the motion comparison measures the combined adaptive process, not an isolated fixed-topology effect. It does not establish a full-training scaling law or multi-scene generality.

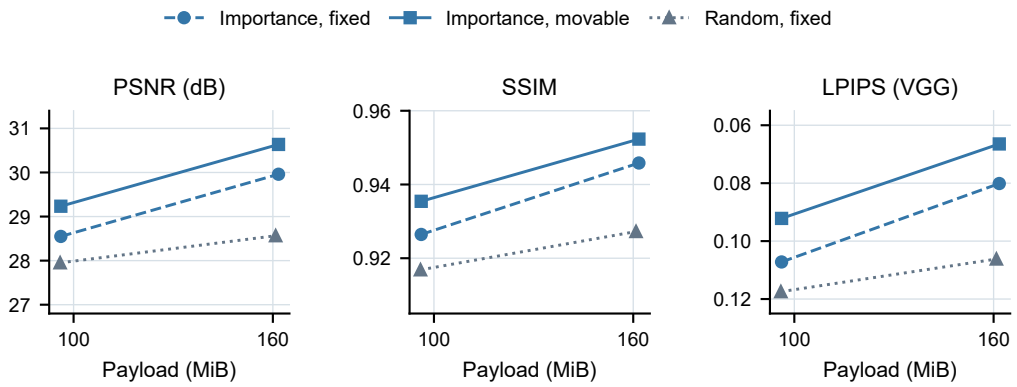

Figure 6: Quality versus persistent tensor payload after equal-length continuation. Importance-guided refinement improves over frozen random allocation at both tested ceilings, while movable vertices improve all three metrics over their frozen-geometry counterparts. The points are independently trained continuation arms, not checkpoints obtained by pruning one model.

D.5 Eight-scene offline LOD protocol and per-scene results

The primary LOD curves start from the eight seed-3721 strict-NVB 30k checkpoints with 4M active tetrahedra each. Importance and RandomEvent begin at the same checkpoint and merge complete legal inverse events at six requested budgets (100/90/80/70/60/50%) that form a branching family: the 90% tier (3.6M target tets) and the 80% tier are merged directly from the full mesh, the 70% tier (2.8M) from the 80% state, the 60% tier (2.4M) from the 70% state, and the 50% tier from the 80% state. Each wave selects round$(0.5m_e)$ of the currently eligible events until the requested budget is reached or no legal events remain. Importance uses absolute gradients on the cell subdivision parameter from 32 uniformly spaced training views rendered at half scale, then takes the maximum score within each complete event; RandomEvent assigns independent seeded uniform scores to complete events. Importance recomputes gradients at each tier. Neither selector uses test images or recovery

optimization. Actual retained fractions differ slightly because events are indivisible. All 88 RevTet states have valid strict certificates and zero dropped-entry, folded-exit, walk-death, and iteration-limit counts over 200 test views.

For SVRaster (Sun et al., 2025), all eight 20k checkpoints use the Synthetic-NeRF configuration. A separate evaluation adapter recomputes the native maximum sample-weight score over training cameras, keeps the highest-scoring voxels to an exact target count, and calls the native pruning operation at the same six budgets. These are post-training pruning states, not independently retrained budget models or an official SVRaster LOD benchmark. The four previously available final checkpoints and four locally reproduced final checkpoints use the same 20k schedule and model/rendering logic. The 48 SVRaster states in these tables have 200 PNG-evaluated test views; the separately reported low-budget tail down to 5% is described below. The plotted budget is each method's own active-primitive fraction: tetrahedra and voxels differ in type and baseline count.

Tables 9–11 list all eight scenes at the six shared target budgets. The mean actual retained fractions are 89.58/79.84/69.73/59.85/49.90% for Importance and 89.72/79.62/69.79/59.65/49.91% for RandomEvent across tiers 90–50%; SVRaster reaches its targets exactly. Differences between the two RevTet arms' scene-mean retained fractions are at most 0.224 percentage points at any tier, so tier-level advantages are descriptive rather than exact equal-count causal effects. The RevTet persistent tensor inventory, including reversible hierarchy, averages 852.41 MiB at full and 801.87/755.75/708.10/661.65/615.00 MiB for Importance and 802.75/755.05/708.73/660.97/615.24 MiB for RandomEvent at the 90–50% tiers; this is not a minimal render-only export or a directly matched SVRaster storage measure.

Table 9: PSNR ↑ (dB). Eight-scene offline LOD on 200 PNG test views per scene. Columns give requested fractions of each method's own full active-primitive count; the exact achieved percentages are listed separately. Tetrahedra and voxels are different units.

Scene	100%	90%	80%	70%	60%	50%
NVB Importance						
Chair	35.59	35.47	35.19	34.71	33.92	32.56
Drums	25.84	25.83	25.79	25.72	25.59	25.29
Ficus	34.70	34.68	34.64	34.53	34.29	33.60
Hotdog	37.54	37.48	37.41	37.22	36.84	36.09
Lego	35.18	35.08	34.82	34.36	33.62	32.05
Materials	29.87	29.84	29.79	29.69	29.51	29.14
Mic	35.13	35.11	35.06	34.92	34.66	33.94
Ship	30.10	30.05	29.95	29.80	29.54	29.14
Mean	32.99	32.94	32.83	32.62	32.25	31.48
RandomEvent						
Chair	35.59	35.32	35.01	34.56	33.88	32.82
Drums	25.84	25.81	25.77	25.69	25.56	25.32
Ficus	34.70	34.68	34.61	34.49	34.21	33.62
Hotdog	37.54	37.38	37.18	36.88	36.50	35.82
Lego	35.18	34.89	34.46	33.90	33.02	31.50
Materials	29.87	29.82	29.75	29.63	29.44	29.10
Mic	35.13	35.08	34.97	34.82	34.53	33.95
Ship	30.10	29.72	29.31	28.84	28.17	27.45
Mean	32.99	32.84	32.63	32.35	31.91	31.20
SVRaster native						
Chair	35.96	35.89	35.57	34.69	33.01	30.72
Drums	26.09	26.04	25.85	25.62	25.28	24.63
Ficus	34.35	34.33	33.99	32.56	29.26	25.57
Hotdog	37.48	37.28	36.67	35.52	33.59	30.53
Lego	35.58	35.46	34.81	33.25	30.75	27.79
Materials	30.01	29.93	29.50	28.49	26.89	24.85
Mic	36.06	35.97	35.44	34.10	32.34	30.37
Ship	30.46	30.42	30.24	29.82	28.96	27.38
Mean	33.25	33.16	32.76	31.76	30.01	27.73

Table 10: SSIM ↑. Eight-scene offline LOD on 200 PNG test views per scene. Columns give requested fractions of each method's own full active-primitive count; the exact achieved percentages are listed separately. Tetrahedra and voxels are different units.

Scene	100%	90%	80%	70%	60%	50%
NVB Importance						
Chair	0.9863	0.9856	0.9842	0.9818	0.9775	0.9692
Drums	0.9443	0.9436	0.9424	0.9407	0.9379	0.9329
Ficus	0.9854	0.9853	0.9850	0.9845	0.9834	0.9807
Hotdog	0.9841	0.9838	0.9833	0.9823	0.9804	0.9765
Lego	0.9809	0.9800	0.9785	0.9757	0.9714	0.9618
Materials	0.9550	0.9545	0.9539	0.9528	0.9510	0.9478
Mic	0.9908	0.9906	0.9904	0.9899	0.9891	0.9870
Ship	0.8885	0.8870	0.8846	0.8811	0.8756	0.8674
Mean	0.9644	0.9638	0.9628	0.9611	0.9583	0.9529
RandomEvent						
Chair	0.9863	0.9850	0.9837	0.9814	0.9777	0.9712
Drums	0.9443	0.9434	0.9420	0.9403	0.9375	0.9332
Ficus	0.9854	0.9853	0.9850	0.9845	0.9834	0.9812
Hotdog	0.9841	0.9835	0.9826	0.9814	0.9797	0.9766
Lego	0.9809	0.9794	0.9772	0.9742	0.9692	0.9599
Materials	0.9550	0.9543	0.9535	0.9525	0.9506	0.9478
Mic	0.9908	0.9906	0.9902	0.9897	0.9888	0.9869
Ship	0.8885	0.8838	0.8783	0.8718	0.8630	0.8515
Mean	0.9644	0.9632	0.9616	0.9595	0.9562	0.9511
SVRaster native						
Chair	0.9863	0.9857	0.9841	0.9807	0.9752	0.9665
Drums	0.9468	0.9459	0.9435	0.9390	0.9312	0.9203
Ficus	0.9844	0.9840	0.9822	0.9774	0.9701	0.9629
Hotdog	0.9842	0.9825	0.9788	0.9719	0.9598	0.9406
Lego	0.9807	0.9797	0.9763	0.9692	0.9568	0.9379
Materials	0.9543	0.9534	0.9505	0.9451	0.9362	0.9231
Mic	0.9919	0.9912	0.9886	0.9840	0.9786	0.9726
Ship	0.8857	0.8836	0.8792	0.8709	0.8566	0.8347
Mean	0.9643	0.9633	0.9604	0.9548	0.9456	0.9323

Table 11: Corrected LPIPS ↓. Eight-scene offline LOD on 200 PNG test views per scene. Columns give requested fractions of each method's own full active-primitive count; the exact achieved percentages are listed separately. Tetrahedra and voxels are different units.

Scene	100%	90%	80%	70%	60%	50%
NVB Importance						
Chair	0.0200	0.0212	0.0235	0.0276	0.0340	0.0453
Drums	0.0623	0.0639	0.0662	0.0696	0.0742	0.0818
Ficus	0.0184	0.0187	0.0192	0.0202	0.0220	0.0260
Hotdog	0.0292	0.0305	0.0321	0.0348	0.0392	0.0472
Lego	0.0220	0.0234	0.0260	0.0306	0.0374	0.0519
Materials	0.0505	0.0516	0.0531	0.0554	0.0589	0.0646
Mic	0.0116	0.0119	0.0126	0.0138	0.0162	0.0208
Ship	0.1387	0.1414	0.1448	0.1493	0.1557	0.1634
Mean	0.0441	0.0453	0.0472	0.0502	0.0547	0.0626
RandomEvent						
Chair	0.0200	0.0217	0.0239	0.0275	0.0333	0.0430
Drums	0.0623	0.0644	0.0670	0.0705	0.0756	0.0825
Ficus	0.0184	0.0188	0.0193	0.0203	0.0221	0.0255
Hotdog	0.0292	0.0312	0.0338	0.0373	0.0425	0.0496
Lego	0.0220	0.0243	0.0284	0.0337	0.0423	0.0570
Materials	0.0505	0.0521	0.0542	0.0565	0.0602	0.0652
Mic	0.0116	0.0122	0.0133	0.0151	0.0182	0.0237
Ship	0.1387	0.1457	0.1531	0.1617	0.1732	0.1871
Mean	0.0441	0.0463	0.0491	0.0528	0.0584	0.0667
SVRaster native						
Chair	0.0170	0.0187	0.0210	0.0247	0.0303	0.0387
Drums	0.0513	0.0523	0.0543	0.0586	0.0655	0.0746
Ficus	0.0166	0.0174	0.0198	0.0241	0.0297	0.0357
Hotdog	0.0256	0.0346	0.0445	0.0557	0.0688	0.0844
Lego	0.0198	0.0212	0.0248	0.0315	0.0421	0.0577
Materials	0.0463	0.0477	0.0510	0.0567	0.0654	0.0776
Mic	0.0090	0.0112	0.0154	0.0205	0.0260	0.0320
Ship	0.1237	0.1250	0.1275	0.1319	0.1390	0.1493
Mean	0.0387	0.0410	0.0448	0.0505	0.0583	0.0687

Table 12: Achieved active-primitive fraction (%). Eight-scene offline LOD on 200 PNG test views per scene. Columns give requested fractions of each method's own full active-primitive count; the exact achieved percentages are listed separately. Tetrahedra and voxels are different units.

Scene	100%	90%	80%	70%	60%	50%
NVB Importance						
Chair	100.00	89.84	80.00	69.69	59.86	49.98
Drums	100.00	89.47	79.96	69.88	59.81	49.88
Ficus	100.00	89.55	79.53	69.55	59.79	49.98
Hotdog	100.00	89.36	79.96	69.91	59.93	49.88
Lego	100.00	89.59	79.85	69.55	59.79	49.91
Materials	100.00	89.84	79.92	69.70	59.75	49.82
Mic	100.00	89.26	79.61	69.65	59.91	49.81
Ship	100.00	89.75	79.93	69.91	59.94	49.95
Mean	100.00	89.58	79.84	69.73	59.85	49.90
RandomEvent						
Chair	100.00	89.28	79.66	69.74	59.70	49.89
Drums	100.00	89.90	79.70	69.86	59.74	49.96
Ficus	100.00	89.91	79.94	69.95	59.60	49.87
Hotdog	100.00	89.86	79.90	69.93	59.55	49.97
Lego	100.00	90.00	79.46	69.60	59.73	49.76
Materials	100.00	89.55	79.45	69.88	59.51	50.00
Mic	100.00	89.74	79.38	69.73	59.75	49.90
Ship	100.00	89.51	79.48	69.61	59.64	49.96
Mean	100.00	89.72	79.62	69.79	59.65	49.91
SVRaster native						
Chair	100.00	90.00	80.00	70.00	60.00	50.00
Drums	100.00	90.00	80.00	70.00	60.00	50.00
Ficus	100.00	90.00	80.00	70.00	60.00	50.00
Hotdog	100.00	90.00	80.00	70.00	60.00	50.00
Lego	100.00	90.00	80.00	70.00	60.00	50.00
Materials	100.00	90.00	80.00	70.00	60.00	50.00
Mic	100.00	90.00	80.00	70.00	60.00	50.00
Ship	100.00	90.00	80.00	70.00	60.00	50.00
Mean	100.00	90.00	80.00	70.00	60.00	50.00

D.6 Observed lower-budget endpoints

SVRaster's native pruning adapter produces valid 200-view renders at all seven tested fractions, down to 5%; its eight-scene mean PSNR falls from 33.25 dB at full to 27.73 dB at 50% and 14.43 dB at 5% (Fig. 7). Rendering at 5% establishes a tested structural state, not a useful-quality threshold or an absolute pruning limit.

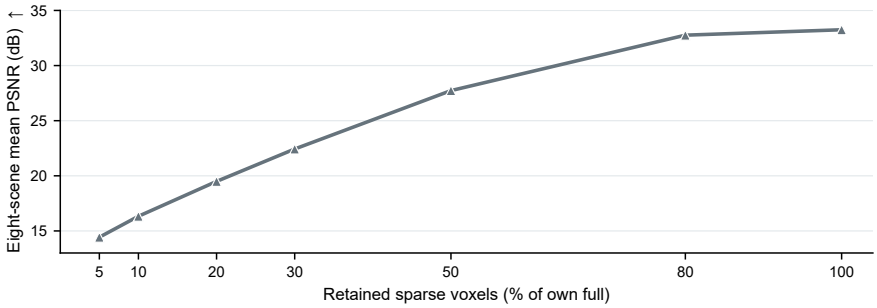

Figure 7: Eight-scene mean SVRaster PSNR under post-training native maximum-weight voxel pruning. The horizontal axis is retained sparse-voxel fraction; no RevTet eight-scene mean exists below 50% under the stopped campaign.

The additional strict-NVB campaign was stopped after four fully certified 200-view endpoints in Chair and Drums. For each path, the 30% request remained unmet because the current greedy path had no more eligible complete inverse events; the 1,600-wave cap was not reached. This is a path-specific terminal state, not a proof of the globally minimum tetrahedron count. No below-50% eight-scene RevTet average is reported, and incomplete states are excluded.

Table 13: Certified strict-NVB path endpoints below 50%; the requested 30% tier was not reached. Corrected LPIPS uses the common 200-view PNG protocol.

Scene	Selection	Tets	Retained	PSNR↑	LPIPS↓
Chair	Importance	1,581,095	39.527%	30.539	.06281
Chair	RandomEvent	1,580,624	39.516%	30.537	.06284
Drums	Importance	1,405,775	35.144%	23.900	.11203
Drums	RandomEvent	1,406,013	35.150%	23.901	.11194