TRISPLAT: SIMULATION-READY FEED-FORWARD 3D SCENE RECONSTRUCTION

Anonymous authors
Paper under double-blind review



Figure 1: **TriSplat reconstructs simulation-ready 3D scenes from sparse, unposed images.** The feed-forward model predicts a triangle mesh in a single pass, enabling direct use in physics engines for locomotion, rigid-body dynamics, and robotic grasping. The teaser is rendered with Blender.

ABSTRACT

Sparse-view 3D reconstruction is increasingly addressed with feed-forward networks. Yet most methods use Gaussian primitives whose surfaces are only implicit. Obtaining a mesh for simulation or embodied interaction still requires costly post-hoc extraction such as TSDF fusion or Poisson reconstruction, breaking the feed-forward promise in pose-free settings where structure and pose are estimated jointly. We present TriSplat, a feed-forward network that represents scenes with oriented triangle primitives and exports meshes from a single forward pass, jointly predicting point maps, triangle attributes, camera poses, and optional intrinsics. Triangle orientation is anchored to geometry rather than regressed freely. Normals from the predicted point maps are refined by an image-conditioned head and converted into stable local frames, while a mono-normal bootstrap, opacity temperature and blur scheduling stabilize early training and sharpen the surface. Experiments on RealEstate10K and DL3DV show more geometry-faithful reconstructions than Gaussian baselines and competitive novel-view rendering. Because the primitives are surface triangles, the output loads directly into physics engines, collision detectors, and rasterization pipelines without conversion, making it a practical simulation-ready solution for feed-forward 3D scene reconstruction.

1 INTRODUCTION

Reconstructing 3D scenes from images is a long-standing problem in computer vision. For robotics and embodied perception (Qureshi et al., 2024; Chhablani et al., 2025), reconstructed scenes must support collision checking, contact-rich planning, and scene-level semantic understanding. Simulation, game, and path-tracing engines build on triangle meshes, so simulation-ready reconstruction must produce explicit meshes that these engines can ingest directly. Classical multi-view pipelines (Schönberger & Frahm, 2016; Schönberger et al., 2016; Furukawa et al., 2015) need multi-stage optimization and calibration, and degrade on sparse views or unknown poses.

Recent feed-forward models (Yu et al., 2021; Chen et al., 2021; 2024b; Zhang et al., 2025b) sidestep per-scene optimization by predicting geometry and rendering primitives directly from images. Gaus-

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.

sian splatting methods (Kerbl et al., 2023; Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025a; Wang et al., 2025b) give efficient high-quality novel views, and pose-free models (Wang et al., 2024; Zhang et al., 2025c; Wang et al., 2025a; Ye et al., 2024; 2025) learn camera estimation and reconstruction jointly. But Gaussian primitives have only implicit surfaces, and point maps have no surface structure at all. Extracting a usable mesh then needs costly post-hoc TSDF fusion or Poisson reconstruction. Geometry-aware variants (Huang et al., 2024; Yu et al., 2024; Lyu et al., 2024; Chang et al., 2025; Dai et al., 2025) still rely on per-scene optimization or auxiliary extraction for meshes. Mesh-generation models such as InstantMesh (Xu et al., 2024), MeshLRM (Wei et al., 2024), and MeshFormer (Liu et al., 2024), and earlier object reconstruction methods (Choy et al., 2016; Kanazawa et al., 2018; Wang et al., 2018), directly predict meshes, yet target object-level reconstruction from controlled viewpoints and do not handle unposed, scene-level inputs.

To close this gap, we present TriSplat, a scene-level simulation-ready feed-forward model whose native representation is a set of oriented triangle primitives. Unlike object-centric mesh generators that recover closed assets, TriSplat reconstructs the visible surfaces and spatial context of an entire scene from sparse, unposed images. This representation directly supports static and kinematic interaction, collision checking, terrain navigation, contact queries, and rasterized rendering, which typically rely on explicit scene surfaces rather than watertight volumes. Watertight object assets can be introduced as additional elements within the scene when more complex physical interactions are required.

Our design follows three observations: (i) for simulation readiness, the rendering primitive itself must be a surface element—triangles satisfy this by construction and can be exported as a mesh without any intermediate extraction; (ii) triangle orientation should be anchored to predicted local geometry rather than learned as an unconstrained variable, providing a strong prior that improves surface fidelity; and (iii) triangles are more sensitive to orientation errors than Gaussian splats, making explicit normal bootstrapping and validity-aware training essential. As illustrated in Fig. 2, TriSplat takes unposed images and, in a single forward pass, predicts point maps, triangle attributes, and camera poses. Geometry normals are refined by an image-conditioned normal head, warm-started from a monocular teacher, and converted into local tangent frames that orient each triangle. Each primitive is then rendered with a differentiable triangle rasterizer and sharpened from soft footprints into crisp, exportable surface elements for downstream physics and graphics pipelines.

Experiments on RealEstate10K and DL3DV show TriSplat surpasses Gaussian feed-forward baselines in both mesh-rendering quality and surface accuracy. Their substantial primitive-rendering advantage does not survive the TSDF conversion their export requires.

Our main contributions are summarized as follows: First, TriSplat is a feed-forward network whose native representation is oriented triangle primitives, jointly predicting geometry, appearance, and poses from sparse unposed images in one pass. Second, a normal-anchored construction derives orientation from predicted point-map geometry and stabilizes training through mono-normal bootstrapping and validity masking. Third, the triangle-native representation removes post-hoc mesh extraction. The output is a directly exportable triangle mesh compatible with physics engines, collision detectors, and rasterization pipelines, and Sec. 3.4 lists the three lightweight export operations.

2 Related Work

Splatting-Based Scene Representations. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) uses anisotropic Gaussian primitives rendered through differentiable alpha-blending. Extensions improve appearance, structure, efficiency, or compression (Zhang et al., 2024; Jiang et al., 2024; Lu et al., 2024; Peng et al., 2024; Zhao et al., 2024; Lee et al., 2024; Chen et al., 2024a; Fan et al., 2024c; Niedermayr et al., 2024), yet volumetric 3D Gaussians still yield view-inconsistent depth and poorly defined surfaces. 2DGS (Huang et al., 2024) collapses each Gaussian to a planar disk suitable for TSDF-based mesh extraction. Gaussian Opacity Fields (Yu et al., 2024), 3DGSR (Lyu et al., 2024), SurfaceSplat (Gao et al., 2025), and related variants (Fan et al., 2024a; Wolf et al., 2024; Cheng et al., 2024; Ververas et al., 2024) couple Gaussians with implicit, stereo, or surface fields for marching-cubes recovery, yet the primitives remain Gaussian and meshes require auxiliary post-processing. Triangle Splatting (Held et al., 2025) replaces Gaussians with oriented triangles rendered through a differentiable rasterizer and produces an exportable mesh, but operates only under per-scene optimization. Concurrent triangle methods (Hardy et al., 2026; Jiang et al., 2026) likewise stay outside the feed-forward pose-free regime.

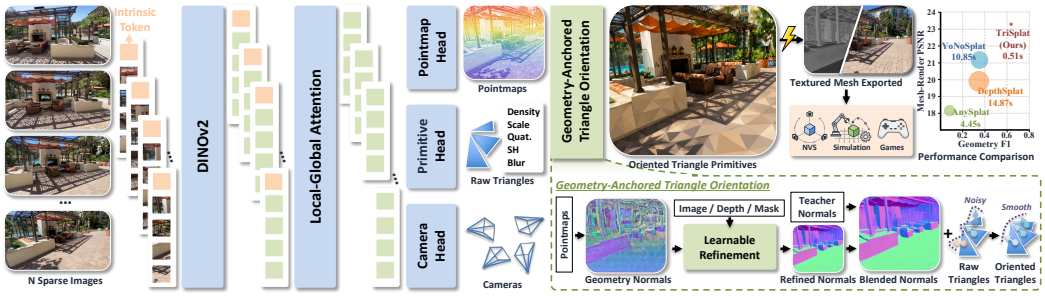


Figure 2: **Overview of TriSplat.** A DINOv2 backbone and Local-Global Attention decoder feed heads that predict point maps, camera poses, and raw triangle attributes. The dashed inset of Sec. 3.2 shows how finite-difference geometry normals are refined by a U-Net, optionally blended with monocular teacher normals (Eftekhar et al., 2021), and converted into tangent frames that orient the triangles. The oriented triangles are rasterized differentiably and exported directly as a textured mesh. The top-right inset plots mesh-rendering PSNR against mesh geometry F1 on 6-view RealEstate10K, where bubble area is proportional to runtime and up-and-right with smaller bubbles is better.

Feed-Forward Sparse-View Reconstruction. Feed-forward methods learn scene priors from large-scale data to predict 3D representations in a single forward pass. Early image-based and NeRF-based approaches (Yu et al., 2021; Su et al., 2024) regress radiance fields from few images but inherit costly volumetric rendering. With 3DGS, explicit feed-forward methods (Wang et al., 2026a; Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025a; Wewer et al., 2024; Zhang et al., 2025a; Min et al., 2024; Tang et al., 2024a; Fei et al., 2024; Jiang et al., 2025; Wang et al., 2025d; Huang et al., 2025; Xiao et al., 2025; Wang et al., 2026b; 2025b; Liu et al., 2025) predict per-pixel Gaussians for efficient, high-quality novel-view synthesis from sparse inputs. A parallel line eliminates known camera poses. DUSt3R (Wang et al., 2024), MASt3R (Leroy et al., 2024; Cabon et al., 2025), VGGT (Wang et al., 2025a), and related models (Yang et al., 2025; Tang et al., 2024b; Keetha et al., 2025; Jang et al., 2025) predict dense geometry to jointly recover structure and relative pose. Their point-map outputs can serve as geometric backbones for either a Gaussian head or a triangle head with an associated surface-construction procedure. In contrast, NoPoSplat (Ye et al., 2024), InstantSplat (Fan et al., 2024b), Splatt3R (Smart et al., 2024), FreeSplatter (Xu et al., 2025b), RegGS (Cheng et al., 2025), UFV-Splatter (Fujimura et al., 2025), FLARE (Zhang et al., 2025c), and YoNoSplat (Ye et al., 2025) predict Gaussian primitives directly in the pose-free setting.

Surface-Aware Feed-Forward Reconstruction. Recent efforts combine feed-forward prediction with stronger surface representations. MeshSplat (Chang et al., 2025) predicts 2DGS through a normal-prediction network and regularizes positions through weighted Chamfer loss. SurfelSplat (Dai et al., 2025) introduces Nyquist-guided surfel adaptation. Both methods retain Gaussian-family primitives and still rely on TSDF fusion for meshes. Our method brings the triangle primitive into the feed-forward, pose-free regime, where oriented triangles used for differentiable rendering can be directly exported as a mesh without auxiliary extraction or per-scene tuning.

3 Method

Given a sparse set of V unposed images, TriSplat reconstructs the scene as oriented triangle primitives in a single forward pass, jointly predicting dense local 3D point maps, per-pixel triangle attributes, camera poses, and optionally intrinsics. Sec. 3.1 describes image-to-primitive prediction, Sec. 3.2 anchors triangle orientation to the predicted geometry, Sec. 3.3 presents the progressive sharpening curriculum, and Sec. 3.4 gives the training objectives and mesh export. Fig. 2 overviews the pipeline.

3.1 From Images to Triangle Primitives

The encoder builds on a DINOv2 (Oquab et al., 2023) backbone with a transformer decoder (Ye et al., 2025). Implementation and architecture details are in Appendix A.

$$\mathbf{p} = z \cdot (u,\ v,\ 1)^\top. \tag{1}$$

This parameterization mirrors the projective image-formation model. The camera head predicts one SE(3) pose per view by mean-pooling decoder tokens and SVD-orthogonalized rota-

tion (Levinson et al., 2020), expressed relative to the first view, with scheduled sampling (Bengio et al., 2015) during training. The primitive head predicts density, scale, quaternion, spherical-harmonic appearance, and blur per pixel, with RGB fused into its features and pixel-shuffle (Shi et al., 2016) upsampling. Head dimensions are in Appendix A.1.

Point maps and poses define triangle centers c in world space. Each triangle is instantiated from a canonical equilateral template, scaled from depth and the pixel footprint, and oriented by the tangent frame R_n of Sec. 3.2 and camera rotation R_c through $v_k = R_c R_n (T_k \odot s) + c$ (Eq. 2), where $\odot$ denotes element-wise multiplication.

$$\mathbf{v}_k = \mathbf{R}_c \, \mathbf{R}_n \big(\mathcal{T}_k \odot \mathbf{s}\big) + \mathbf{c}, \quad k \in \{1, 2, 3\}, \tag{2}$$

A differentiable triangle rasterizer (Held et al., 2025) renders the triangles with tile-based sorting and front-to-back alpha compositing, producing RGB, depth, and normals. Code-level construction details are in Appendix E.

The point maps produced by this stage serve a dual purpose. Beyond defining triangle centers, they also provide the geometric foundation for deriving triangle orientation, as we describe next.

3.2 Anchoring Triangle Orientation to Geometry

Triangle primitives are far more orientation-sensitive than Gaussian splats. A misoriented Gaussian leaves a plausible soft footprint, whereas a misoriented triangle creates hard-edged artifacts whose visibility grows with angular error, especially when the target viewpoint differs substantially from the input views. Orientation is therefore anchored to predicted 3D geometry rather than learned as a free latent variable.

Geometry normals. Finite differences of the point map yield the raw geometry normal $n_{geo} = \mathrm{normalize}(\Delta_x \times \Delta_y)$, flipped toward the camera and defined only on a validity mask propagated through later stages. Optional detach, smoothing, and an orientation-aware box filter suppress high-frequency noise while preserving depth discontinuities, as detailed in Appendix A.1.

$$\mathbf{n}_{\mathrm{geo}} = \mathrm{normalize}(\Delta_x \times \Delta_y), \tag{3}$$

which flips toward the camera when $\mathbf{n}_{\mathrm{geo}} \cdot \mathbf{p} > 0$. Border pixels and degenerate cross products are excluded via a Boolean validity mask $\mathbf{m}$ propagated through all subsequent stages. The point map may be detached to decouple normal refinement from point prediction, and high-frequency noise is suppressed by average-pool smoothing plus an orientation-aware box filter that weights only sign-agreeing neighbors and preserves depth-edge discontinuities.

These geometry normals provide a strong structural prior but are inevitably noisy during early training when point maps have not converged. Two complementary mechanisms address this.
Learned refinement. A lightweight zero-initialized U-Net refines the normals with appearance and depth cues. It takes the raw and smoothed geometry normals, the RGB image, the predicted depth, and the validity mask, and starts as an identity mapping that gradually learns corrections, $n_{ref} = \mathrm{normalize}(n_{sm} + f_\theta(\cdot))$ (Eq. 4). Architecture details are in Appendix A.1.

$$\mathbf{n}_{\mathrm{ref}} = \mathrm{normalize}\big(\mathbf{n}_{\mathrm{sm}} + f_\theta(\mathbf{n}_{\mathrm{geo}}, \, \mathbf{n}_{\mathrm{sm}}, \, \mathbf{I}_v, \, \mathbf{D}_v, \, \mathbf{m})\big). \tag{4}$$

Zero initialization is critical for stability, since a random head would perturb orientations before useful gradients accumulate.

Mono-normal bootstrap. Early training is a chicken-and-egg problem. Point maps are too inaccurate for normals and the refinement head has not learned corrections. A bootstrap schedule warm-starts orientation from a pretrained monocular normal estimator (Eftekhar et al., 2021) and blends teacher normals with model normals through $n_{fwd} = \mathrm{normalize}(\alpha n_{tch} + (1 - \alpha) n_{ref})$ (Eq. 5). Teacher takeover, cosine-decayed blending, and full release follow Eq. 6 and the schedule in Appendix A.4, only where both validity masks hold.

$$\mathbf{n}_{\mathrm{fwd}} = \mathrm{normalize}\big(\alpha(t) \, \mathbf{n}_{\mathrm{tch}} + (1 - \alpha(t)) \, \mathbf{n}_{\mathrm{ref}}\big). \tag{5}$$

The schedule comprises three phases: a *takeover* phase ($t \le t_{\mathrm{tk}}$, $\alpha = 1$) where the teacher fully determines orientation; a *blending* phase ($t_{\mathrm{tk}} < t < t_{\mathrm{bl}}$) where α decays via a cosine schedule

$$\alpha(t) = \tfrac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{t - t_{\mathrm{tk}}}{t_{\mathrm{bl}} - t_{\mathrm{tk}}}\right)\right); \tag{6}$$

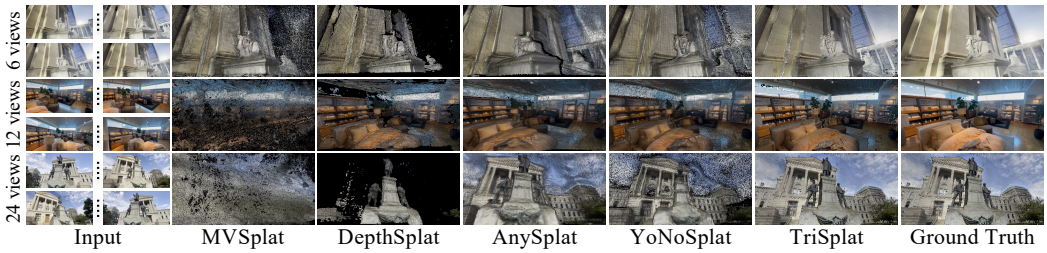


Figure 3: **Mesh-rendering comparison on DL3DV.** Rows are 6, 12, and 24 input views. After mesh export, Gaussian baselines show missing surfaces and inconsistent geometry, whereas TriSplat keeps more complete triangle-rendered structure in novel views.

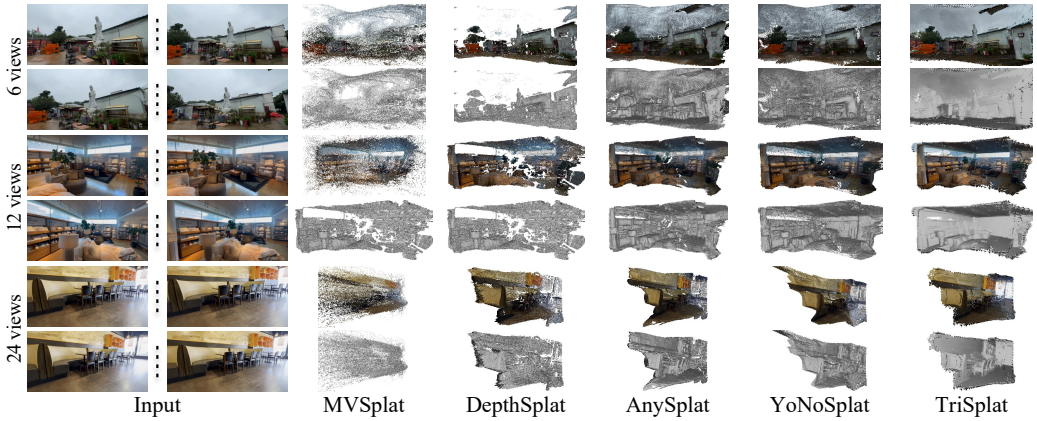


Figure 4: **Textured mesh comparison on DL3DV.** 6/12/24 input views. Gaussian-to-TSDF conversion shrinks or fragments geometry and loses scene extent. TriSplat exports more coherent textured triangle surfaces for downstream rendering.

and a *release* phase ($t \geq t_{\text{bl}}$, $\alpha = 0$) where the model relies entirely on its own geometry. Blending is restricted to pixels where both teacher and geometry validity masks hold. This bootstrap acts on the forward-pass representation rather than as a loss term: the teacher normal enters triangle construction and shapes the rendered output and all downstream gradients. We apply both mechanisms throughout early training.

Tangent frame construction. The blended normal defines a full orthonormal tangent frame whose rotation serves directly as R_n in Eq. 2 at valid pixels. Construction details are in Appendix E.

3.3 Progressive Surface Sharpening

Oversized or misplaced Gaussians still cover roughly the correct region and receive gradients. A triangle in the same situation may miss its target pixels and produce zero gradients. We schedule two softness parameters that transition primitives from blurred and forgiving to sharp and mesh-ready.

Opacity temperature. The encoder output opacity $p \in [0, 1]$ is passed unchanged to the triangle representation. During differentiable rendering, we apply a scheduled temperature to sharpen triangle opacity values before rasterization:

$$\tilde{p}(t) = \sigma(\tau(t)\,\text{logit}(\text{clip}(p, \epsilon, 1-\epsilon)))\,, \qquad \epsilon = 10^{-6}. \tag{7}$$

A low initial temperature keeps triangle opacities soft and partially transparent, allowing more triangles to receive useful gradients and stabilizing early optimization. As τ increases, opacities approach binary values, making the representation more mesh-like and reducing rendering loss after direct mesh export. The schedule is given in Appendix D.

Blur scheduling. Large initial blur creates broad overlapping footprints with dense gradients. As β decays, each triangle tightens into a defined surface element. Schedules are in Appendix A.3.

$$\sigma = \text{sigmoid}(\hat{\sigma}) \cdot \beta(t), \tag{8}$$

where $\hat{\sigma}$ is the raw value and $\beta(t)$ decays over training. Large early blur gives broad footprints with dense gradients, and each triangle tightens as β decays. Appendix A.3 gives the schedule.

Table 1: **Surface quality on DL3DV with 6/12/24 input views.** Gaussian baselines use TSDF-fused meshes and the triangle-native model exports directly. Best results in **bold**, second-best underlined.

Method	6 views				12 views				24 views			
	CD↓	Prec.↑	Rec.↑	F1↑	CD↓	Prec.↑	Rec.↑	F1↑	CD↓	Prec.↑	Rec.↑	F1↑
MVSplat	1.143	0.121	0.154	0.118	0.802	0.122	0.211	0.135	0.695	0.130	0.276	0.156
DepthSplat	1.116	0.135	0.192	0.145	0.907	0.127	0.237	0.152	0.786	0.120	0.274	0.152
AnySplat	1.012	0.088	0.143	0.093	0.731	0.085	0.163	0.096	0.699	0.084	0.175	0.100
YoNoSplat	0.920	0.090	0.189	0.106	0.664	0.073	0.183	0.092	0.687	0.068	0.173	0.088
TriSplat (Ours)	**0.613**	**0.223**	**0.448**	**0.287**	**0.323**	**0.200**	**0.522**	**0.279**	**0.310**	**0.191**	**0.571**	**0.277**

Table 2: **NVS quality on DL3DV with 6/12/24 input views under mesh rendering.** Target-view quality of exported meshes rendered with a single triangle rasterizer.

Method	6 views			12 views			24 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVSplat	14.75	0.450	0.509	15.16	0.431	0.525	15.72	0.439	0.526
DepthSplat	14.86	0.481	0.490	14.82	0.426	0.536	15.13	0.422	0.541
AnySplat	18.58	0.551	0.397	16.58	0.485	0.462	16.42	0.442	0.475
YoNoSplat	18.88	0.548	0.414	16.90	0.459	0.485	16.71	0.452	0.499
TriSplat (Ours)	**20.84**	**0.615**	**0.335**	**18.71**	**0.493**	**0.400**	**18.06**	**0.455**	**0.425**

3.4 Training Objectives and Mesh Extraction

Training objectives. TriSplat is trained end-to-end with three terms for rendering, camera, and orientation:

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{cam}} + \mathcal{L}_{\text{normal}}. \tag{9}$$

The photometric term combines pixel-wise reconstruction with a perceptual LPIPS loss (Zhang et al., 2018). The camera term is a pairwise relative-pose loss over ordered view pairs, invariant to the global frame. The normal term aligns refined normals with monocular teacher normals at valid pixels. Appendix A.2 gives the formulations, weights, and large-loss filter.

Mesh extraction. Mesh export requires no auxiliary reconstruction. After a forward pass, low-opacity triangles are discarded, face winding is corrected against the per-pixel normals of Sec. 3.2, and duplicate vertices are merged by quantized position hashing. These three lightweight export operations replace per-scene optimization, TSDF fusion, or marching cubes. They yield a standard triangle mesh directly usable in mesh-based physics simulation, collision detection, and rendering engines. Appendix B lists the export parameters.

4 Experiments

We evaluate TriSplat along four axes, namely reconstructed surface geometry, novel-view rendering of the exported mesh under a standard triangle rasterizer, zero-shot depth and normal accuracy, and runtime efficiency. Controlled ablations validate the design choices.

4.1 Experimental Setup

Datasets. We train on RE10K (67K scenes) and DL3DV (10K+ scenes) following standard splits (Ye et al., 2025), and evaluate zero-shot on 100 held-out ScanNet scenes following MeshSplat (Chang et al., 2025). Appendix A.3 details the splits.

Baselines. We compare against feed-forward Gaussian methods (Chen et al., 2024b; Xu et al., 2025a; Jiang et al., 2025; Ye et al., 2025; Chang et al., 2025; Dai et al., 2025), namely cost-volume MVSplat and DepthSplat, pose-free AnySplat and YoNoSplat, and geometry-aware MeshSplat and SurfelSplat. DUSt3R (Wang et al., 2024) and VGGT (Wang et al., 2025a) are considered as geometric backbones whose point-map outputs can be combined with either Gaussian or triangle heads. To keep the main tables focused on comparable end-to-end representation pipelines, we do not include these backbone–head combinations as separate baselines.



Figure 5: **Mesh-rendering comparison on RE10K.** Target-view renders from exported meshes with six input views under the same mesh-rendering protocol. TSDF-fused Gaussian baselines blur surfaces or drop structures, while TriSplat preserves sharper detail and more complete silhouettes in novel views.

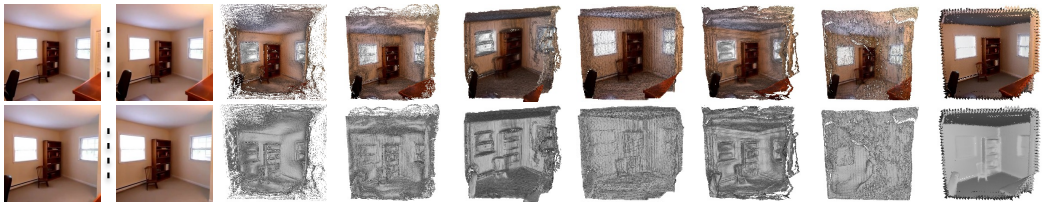


Figure 6: **Textured mesh comparison on RE10K.** Textured and geometry-only views of exported meshes for one representative scene. TriSplat yields cleaner triangle surfaces, whereas TSDF-fused baselines over-smooth or fragment geometry.

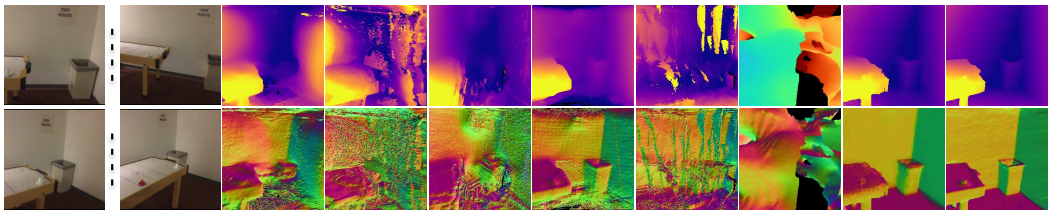


Figure 7: **Zero-shot depth and normal comparison on ScanNet.** Depth and surface-normal predictions from RE10K-trained models. TriSplat produces smoother, surface-aligned normals than TSDF-fused Gaussian baselines.

Metrics. Surface quality uses Chamfer Distance (CD), Precision, Recall, and F1, computed with the protocol detailed in Appendix J. Rendering quality uses PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) on the exported mesh. ScanNet additionally reports depth and normal accuracy in zero-shot evaluation.

Surface evaluation protocol. For geometric evaluation, we reconstruct the reference surface from the corresponding COLMAP dense reconstruction and align it to the dataset coordinate frame with a scene-level similarity transformation before computing the surface metrics.

Evaluation protocol and rendering modes. We evaluate DL3DV with 6, 12, and 24 views at gaps of 50 to 180 frames and RE10K with 6 views at gaps of 50 to 150 frames, and ScanNet zero-shot with RE10K-trained models. Primitive rendering uses the native rasterizer of each method, while mesh rendering uses a standard triangle rasterizer on the exported mesh. Gaussian baselines export through TSDF fusion (Chang et al., 2025), while TriSplat exports triangles directly. Under primitive rendering the strongest Gaussian baseline is competitive with or slightly ahead of TriSplat, with RE10K PSNR 27.19 versus 26.46 and DL3DV 23.88 versus 23.22. Since the target is a mesh for physics engines and graphics, mesh rendering is the primary metric. We report primitive rendering results together with an explicit primitive-to-mesh degradation analysis in Appendix C.

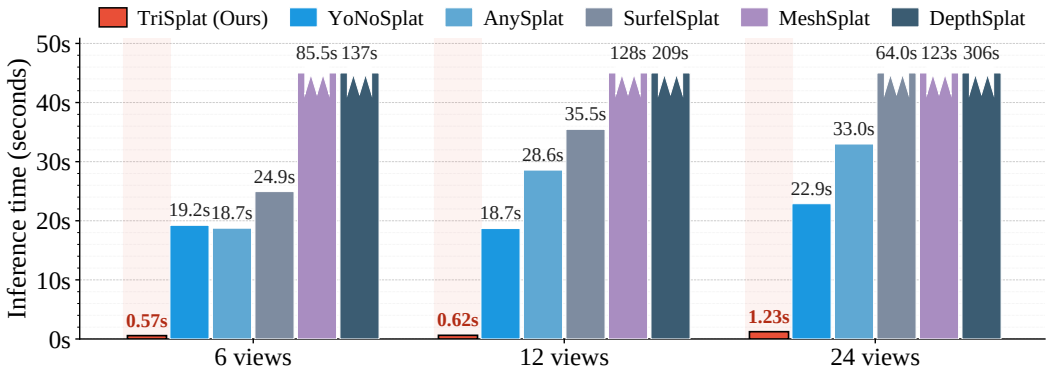


Figure 8: **Runtime comparison on DL3DV.** TriSplat produces a usable mesh in under **1.3** s for up to 24 input views, while Gaussian baselines also pay a volume-dependent mesh-export cost. Bars above 45 s are broken, with exact values annotated above each bar.

Table 3: **Quantitative comparison on RE10K with 6 input views.** Same mesh-based protocol as Tables 1 and 2, including surface-aware MeshSplat (Chang et al., 2025) and SurfelSplat (Dai et al., 2025) that target geometric fidelity rather than rendering.

Method	Surface Quality				NVS Quality		
	CD ↓	Prec. ↑	Rec. ↑	F1 ↑	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat	0.340	0.618	0.257	0.358	13.97	0.615	0.378
DepthSplat	0.294	0.647	0.326	0.429	21.23	0.781	0.271
AnySplat	0.540	0.460	0.064	0.110	18.23	0.589	0.365
YoNoSplat	0.267	0.681	0.333	0.443	21.94	0.753	**0.238**
MeshSplat	0.349	0.642	0.234	0.340	19.97	0.719	0.294
SurfelSplat	0.747	0.432	0.099	0.154	11.18	0.184	0.738
TriSplat (Ours)	**0.190**	**0.708**	**0.560**	**0.622**	**24.69**	**0.798**	0.269

Table 4: **Zero-shot depth and normal evaluation on ScanNet with 6 input views.** Models trained on RE10K Zhou et al. (2018), zero-shot evaluated on ScanNet.

Method	Depth		Normal	
	Rel ↓	Diff ↓	Mean ↓	$< 30^\circ$ ↑
MVSplat	0.708	1.206	102.247	17.204
DepthSplat	0.279	0.595	54.861	29.403
AnySplat	0.453	**0.283**	55.557	25.375
YoNoSplat	0.270	0.516	54.110	41.047
MeshSplat	0.534	0.999	59.803	31.862
SurfelSplat	0.716	1.264	75.300	16.484
TriSplat (Ours)	**0.188**	0.341	**27.901**	**71.708**

4.2 Surface Reconstruction and Mesh Rendering

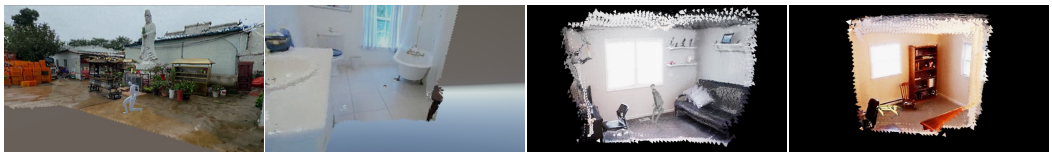
Tables 1, 2, and 3 jointly report the geometric quality of the exported mesh and its novel-view quality under a standard triangle pipeline, measuring both surface faithfulness and performance as a downstream rendering primitive.

Surface geometry. TriSplat produces the most accurate surface geometry across all four metrics on both datasets. On RE10K it attains CD 0.190 and F1 0.622, improving over the strongest Gaussian baseline YoNoSplat by 0.077 in CD and 0.179 in F1. The gap is largest on Recall at +0.227. TSDF-fused meshes systematically under-cover the ground-truth surface, particularly around thin structures. Surface-oriented Gaussian variants do not close this gap. MeshSplat remains bounded by TSDF discretization, and SurfelSplat degrades substantially on CD and F1. The same pattern holds on DL3DV across 6, 12, and 24 views. Qualitative textured-mesh visualizations in Figs. 6 and 4 confirm the trend, with more examples in Appendix I.

Mesh rendering. A standard triangle rasterizer is one important downstream consumer of a simulation-ready representation. We therefore evaluate rendering directly on the exported mesh, in addition to reporting native primitive rendering in the Appendix C. On RE10K TriSplat reaches 24.69 dB mesh-rendering PSNR versus 21.94 dB for the strongest Gaussian baseline, a 2.75 dB margin. The advantage stems from a structural asymmetry. Gaussian baselines incur a substantial quality drop when their TSDF-fused meshes are rendered as triangles, because the discretized volume discards the very primitives that produced the original image. In TriSplat the rendering primitives are the mesh, so no information is lost during export.

4.3 Depth and Normal Quality

Tab. 4 evaluates depth and normal accuracy on ScanNet zero-shot with RE10K-trained models. TriSplat achieves the lowest AbsRel of 0.188 and the second-lowest AbsDiff of 0.341. On normals it reaches 27.9 degrees mean error and 71.7 percent within 30 degrees, against 54.1 degrees and



Unity character | Unity interaction | Isaac H1 | Isaac quadruped

Figure 9: **Simulation-ready demonstration.** Directly exported TriSplat meshes in Unity and NVIDIA Isaac Sim for interaction, locomotion, and collision, reused without Gaussian-to-mesh conversion or scene-specific cleanup.

41.0 percent for the strongest baseline. Qualitative depth and normal maps are in Fig. 7; additional novel-view synthesis results on ScanNet are in Appendix F.

4.4 Efficiency

Fig. 8 compares end-to-end time-to-mesh on DL3DV at 6, 12, and 24 views on a single H100. The TriSplat primitives are themselves the mesh, so the cost is the feed-forward pass alone at 0.57 s, 0.62 s, and 1.23 s. Every Gaussian baseline also runs TSDF fusion scaling with the reconstructed volume. The fastest, AnySplat, takes 18.7 s at 6 views and 33.0 s at 24 views, while DepthSplat reaches 306 s at 24 views. TriSplat is thus 33 times faster than the fastest Gaussian baseline at 6 views and up to 249 times faster than the slowest at 24 views, the only method under one second.

4.5 Simulation-Ready Demonstration

To validate the practical utility of our simulation-ready representation, we load directly exported TriSplat meshes into Unity and NVIDIA Isaac Sim, demonstrating rigid-body simulation, collision detection, navigation, and grasping in Fig. 9. The meshes load without manual cleanup or format conversion, while Gaussian baselines require TSDF fusion and mesh cleaning. Extended simulation experiments, including rigid-body dynamics and legged locomotion, are provided in Appendix H.

4.6 Ablation Study

Tab. 5 ablates four design choices on RE10K with 6 views. Every removal lowers precision and rendering metrics. Without normal anchoring, centers drift, precision drops 0.057, and PSNR falls 2.55 dB. Without the mono-normal bootstrap that breaks this deadlock, surface degradation is largest, with precision down 0.065 and PSNR down 2.52 dB. Without normal refinement, rendering drops most, with PSNR down 3.02 dB and LPIPS up 0.160. Without progressive sharpening and its early soft gradients, PSNR drops 2.88 dB and precision 0.062. Finer-grained hyperparameter ablations are provided in Appendix G.

Table 5: **Ablation study on RE10K (6 views).** One component removed per row. Geometry metrics on the exported mesh, rendering metrics under mesh rendering.

Config.	Surface		NVS	
	CD ↓	Prec. ↑	PSNR ↑	LPIPS ↓
Full model	**0.190**	**0.708**	**24.69**	**0.269**
w/o anchoring	0.190	0.651	22.14	0.396
w/o bootstrap	0.198	0.643	22.17	0.397
w/o refinement	0.193	0.649	21.67	0.429
w/o sharpening	0.191	0.646	21.81	0.416

5 Conclusion

We presented TriSplat, a feed-forward model that represents scenes as oriented triangle primitives and jointly predicts geometry, appearance, and camera parameters from sparse unposed images in one pass. Anchoring orientation to point-map geometry, a mono-normal bootstrap, and a sharpening curriculum yield substantially more accurate surface geometry than Gaussian feed-forward baselines. The rendering primitives are the exported mesh, so TriSplat attains the strongest mesh-rendering quality across RE10K, DL3DV, and zero-shot ScanNet while avoiding the primitive-to-mesh degradation TSDF fusion imposes. The exported meshes can be ingested by mainstream physics engines, making simulation readiness a property of the representation rather than a conversion step.

Limitations and future work. The export produces a non-manifold triangle soup adequate for rigid-body contact, collision, navigation, and rasterized rendering but not for watertight applications. Triangle density follows input resolution, and adaptive tessellation remains future work.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript preparation, including format migration and language editing. The authors reviewed all AI-assisted work and remain responsible for the final text, claims, citations, experimental results, and artifacts in this submission.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the details needed to rebuild and re-evaluate TriSplat in the appendix, and we release the source code for training and evaluation in the supplementary material. The network architecture, including the point, camera, and primitive heads and the refinement U-Net, is specified in Appendix A.1. The complete training objectives and their weights are given in Appendix A.2, and the training protocol, schedules, and initialization are described in Appendix A.3. The mono-normal bootstrap teacher and the teacher-blending schedule are specified in Appendix A.4, the baseline mesh-extraction parameters used for Gaussian baselines in Appendix B, the triangle-adapter construction details in Appendix E, and the mesh-evaluation protocol (point-cloud sampling, distance computation, and metric definitions) in Appendix J. All reported quantitative results in the main paper are accompanied by the data splits, input-view settings, and evaluation mode (primitive versus mesh rendering) used to produce them.

REFERENCES

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems*, 28, 2015.

Yohann Cabon, Lucas Stoffl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. Must3r: Multi-view network for stereo 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1050–1060, 2025.

Hanzhi Chang, Ruijie Zhu, Wenjie Chang, Mulin Yu, Yanzhe Liang, Jiahao Lu, Zhuoyuan Li, and Tianzhu Zhang. Meshsplat: Generalizable sparse-view surface reconstruction via gaussian splatting. *arXiv preprint arXiv:2508.17811*, 2025.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 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 *IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pp. 422–438. Springer, 2024a.

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 *European Conference on Computer Vision*, pp. 370–386. Springer, 2024b.

Chong Cheng, Yu Hu, Sicheng Yu, Beizhen Zhao, Zijian Wang, and Hao Wang. RegGS: Unposed sparse views gaussian splatting with 3DGS registration. In *IEEE/CVF International Conference on Computer Vision*, 2025.

Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *International Conference on Machine Learning*, 2024.

Gunjan Chhablani, Xiaomeng Ye, Muhammad Zubair Irshad, and Zsolt Kira. Embodiedsplat: Personalized real-to-sim-to-real navigation with gaussian splats from a mobile device. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 25431–25441, 2025.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*, pp. 628–644. Springer, 2016.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5828–5839, 2017.

Chensheng Dai, Shengjun Zhang, Min Chen, and Yueqi Duan. Surfelsplat: Learning efficient and generalizable gaussian surfel representations for sparse-view surface reconstruction. In *Advances in Neural Information Processing Systems*, 2025.

Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *IEEE/CVF International Conference on Computer Vision*, pp. 10786–10796, 2021.

Lue Fan, Yuxue Yang, Minxing Li, Hongsheng Li, and Zhaoxiang Zhang. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024a.

Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *CoRR*, 2024b.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in Neural Information Processing Systems*, 37:140138–140158, 2024c.

Xin Fei, Wenzhao Zheng, Yueqi Duan, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Jiwen Lu. Pixelgaussian: Generalizable 3d gaussian reconstruction from arbitrary views, 2024. URL https://arxiv.org/abs/2410.18979.

Yuki Fujimura, Takahiro Kushida, Kazuya Kitano, Takuya Funatomi, and Yasuhiro Mukaigawa. Ufv-splatter: Pose-free feed-forward 3d gaussian splatting adapted to unfavorable views. *arXiv preprint arXiv:2507.22342*, 2025.

Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

Zihui Gao, Jia-Wang Bian, Guosheng Lin, Hao Chen, and Chunhua Shen. Surfacesplat: Connecting surface reconstruction and gaussian splatting. *arXiv preprint arXiv:2507.15602*, 2025.

Romain Hardy, Andrea Dunn Beltran, Todd A. Brenner, Tyler M. Berzin, and Pranav Rajpurkar. Rendering with a gut feeling: Depth-guided triangle splatting for physically consistent colonoscopic reconstruction. In *Medical Imaging with Deep Learning (MIDL)*, pp. 1637–1655, 2026.

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

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conference Proceedings*, pp. 1–11, 2024.

Guichen Huang, Ruoyu Wang, Xiangjun Gao, Che Sun, Yuwei Wu, Shenghua Gao, and Yunde Jia. Longsplat: Online generalizable 3d gaussian splatting from long sequence images. *arXiv preprint arXiv:2507.16144*, 2025.

Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jerome Revaud. Pow3r: Empowering unconstrained 3d reconstruction with camera and scene priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1071–1081, 2025.

Changjian Jiang, Kerui Ren, Linning Xu, Jiong Chen, Jiangmiao Pang, Yu Zhang, Bo Dai, and Mulin Yu. Halogs: Loose coupling of compact geometry and gaussian splats for 3d scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2026. arXiv:2505.20267.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *arXiv preprint arXiv:2505.23716*, 2025.

Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5322–5332, 2024.

Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision*, pp. 371–386, 2018.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

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, 2023.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21719–21728, 2024.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pp. 71–91. Springer, 2024.

Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *Advances in Neural Information Processing Systems*, 33:22554–22565, 2020.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. Meshformer: High-quality mesh generation with 3d-guided reconstruction model. *Advances in Neural Information Processing Systems*, 37: 59314–59341, 2024.

Xinhang Liu, Yuxi Xiao, Donny Y Chen, Jiashi Feng, Yu-Wing Tai, Chi-Keung Tang, and Bingyi Kang. Trace anything: Representing any video in 4d via trajectory fields. *arXiv preprint arXiv:2510.13802*, 2025.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20654–20664, 2024.

Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *ACM Transactions on Graphics*, 43(6):1–12, 2024.

Zhiyuan Min, Yawei Luo, Jianwen Sun, and Yi Yang. Epipolar-free 3d gaussian splatting for generalizable novel view synthesis. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 39573–39596. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/45ed1a72597594c097152ef9cc187762-Paper-Conference.pdf.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10349–10358, 2024.

Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

Cheng Peng, Yutao Tang, Yifan Zhou, Nengyu Wang, Xijun Liu, Deming Li, and Rama Chellappa. Bags: Blur agnostic gaussian splatting through multi-scale kernel modeling. In *European Conference on Computer Vision*, pp. 293–310. Springer, 2024.

Mohammad Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*, 2024.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883, 2016.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024.

Chih-Hai Su, Chih-Yao Hu, Shr-Ruei Tsai, Jie-Ying Lee, Chin-Yang Lin, and Yu-Lun Liu. Boost-mvsnerfs: Boosting mvs-based nerfs to generalizable view synthesis in large-scale scenes. In *ACM SIGGRAPH Conference Proceedings*, pp. 1–12, 2024.

Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction, 2024a. URL `https://arxiv.org/abs/2410.06245`.

Zhenggang Tang, Yuchen Fan, Dilin Wang, Hongyu Xu, Rakesh Ranjan, Alexander Schwing, and Zhicheng Yan. Mv-dust3r+: Single-stage scene reconstruction from sparse views in 2 seconds. *arXiv preprint arXiv:2412.06974*, 2024b.

Evangelos Ververas, Rolandos Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. Sags: structure-aware 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 221–238. Springer, 2024.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. *arXiv preprint arXiv:2503.11651*, 2025a.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision*, pp. 52–67, 2018.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20697–20709, 2024.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. Volsplat: Rethinking feed-forward 3d gaussian splatting with voxel-aligned prediction. *arXiv preprint arXiv:2509.19297*, 2025b.

Weijie Wang, Qihang Cao, Sensen Gao, Donny Y Chen, Haofei Xu, Wenjing Bian, Songyou Peng, Tat-Jen Cham, Chuanxia Zheng, Andreas Geiger, et al. Feed-forward 3d scene modeling: A problem-driven perspective. *arXiv preprint arXiv:2604.14025*, 2026a.

Weijie Wang, Donny Y Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. Zpressor: Bottleneck-aware compression for scalable feed-forward 3dgs. *Advances in Neural Information Processing Systems*, 38:113407–113436, 2026b.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. pi3: Scalable permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025c.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat++: Generalizable 3d gaussian splatting for efficient indoor scene reconstruction. *arXiv preprint arXiv:2503.22986*, 2025d.

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*, 13(4):600–612, 2004.

Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality meshes. *arXiv preprint arXiv:2404.12385*, 2024.

Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *European Conference on Computer Vision*, pp. 456–473. Springer, 2024.

Yaniv Wolf, Amit Bracha, and Ron Kimmel. Surface reconstruction from gaussian splatting via novel stereo views. *arXiv e-prints*, pp. arXiv–2404, 2024.

Yang Xiao, Guoan Xu, Qiang Wu, and Wenjing Jia. Jointsplat: Probabilistic joint flow-depth optimization for sparse-view gaussian splatting. *arXiv preprint arXiv:2506.03872*, 2025.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025a.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

Jiale Xu, Shenghua Gao, and Ying Shan. Freesplatter: Pose-free gaussian splatting for sparse-view 3d reconstruction. In *IEEE/CVF International Conference on Computer Vision*, 2025b.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. *arXiv preprint arXiv:2511.07321*, 2025.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

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

Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. In *AAAI Conference on Artificial Intelligence*, volume 39, pp. 9869–9877, 2025a.

Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21424–21433, 2024.

Jiahui Zhang, Yuelei Li, Anpei Chen, Muyu Xu, Kunhao Liu, Jianyuan Wang, Xiao-Xiao Long, Hanxue Liang, Zexiang Xu, Hao Su, et al. Advances in feed-forward 3d reconstruction and view synthesis: A survey. *arXiv preprint arXiv:2507.14501*, 2025b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21936–21947, 2025c.

Lingzhe Zhao, Peng Wang, and Peidong Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting. In *European Conference on Computer Vision*, pp. 233–250. Springer, 2024.

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

A Implementation Details

A.1 Network Architecture

Backbone. The encoder adopts a DINOv2 ViT-L/14 (Oquab et al., 2023) backbone with patch size 14 and augments it with 2D rotary position embeddings. Decoder blocks alternate between intra-view self-attention and cross-view joint attention, producing feature tokens of dimension $d = 1024$. For pose-free operation the backbone additionally embeds per-pixel intrinsic information via a 4th-degree positional encoding applied at pixel level.

Point and camera heads. The point head is a 5-layer transformer decoder (dimension 1024, 16 heads, MLP ratio 4) followed by a linear projection that outputs three channels (two lateral coordinates and one log-depth), upsampled via pixel-shuffle (Shi et al., 2016) to $14\times$ the token resolution. Each pixel predicts three unconstrained scalars (u, v, z'), and depth is recovered as $z = \exp(z')$, which enforces strict positivity of the predicted depth, with the 3D point given by (u, v, z) in the camera coordinate frame. The camera head shares the same transformer depth and head count but reduces the output dimension to 512, mean-pools the decoded tokens via adaptive average pooling, and maps them through two residual 1×1 convolution blocks and two MLP layers to produce per-view SE(3) poses.

Primitive head. The primitive head is structurally identical to the point head but its input features are additively fused with zero-initialized patch-embedded RGB tokens before decoding, providing the branch with direct access to appearance information. Its output dimension is $1 + d_{\text{tri}}$, where $d_{\text{tri}} = 3 + 4 + 3 + 1 = 11$ consists of three scale logits, a four-component quaternion, three zeroth-order spherical-harmonic (SH) coefficients, and one blur parameter. All dense heads employ an upscale token ratio of 2 and generate predictions at 14×14 points per patch, yielding a dense prediction map at the input image resolution.

Normal refinement U-Net. The geometry-anchored normal refinement head is a lightweight U-Net with 4 encoder–decoder scales. Each scale consists of a convolution stage (conv $\rightarrow$ GroupNorm $\rightarrow$ GELU) followed by 2 residual convolution blocks, all using 3×3 kernels. The channel progression through the encoder is $36 \rightarrow 72 \rightarrow 144 \rightarrow 288$ and is mirrored in the decoder, which upsamples with bilinear interpolation and concatenates skip features. The network receives 11 input channels comprising the raw geometry normal (3), the smoothed geometry normal (3), the downsampled RGB image (3), the predicted depth (1), and the validity mask (1). The output layer operates in residual mode with a scale factor of 0.25, where both weights and biases are zero-initialized so that the head starts as an identity mapping and gradually learns corrections. Training uses mixed-precision bfloat16 with gradient checkpointing enabled to reduce memory.

A.2 Loss Formulation

We expand each term of the training objective $\mathcal{L}$ in Sec. 3.4 of the main paper. With per-term weights, the total objective is

$$\mathcal{L} = \lambda_{\text{photo}}\,\mathcal{L}_{\text{photo}} + \lambda_{\text{cam}}\,\mathcal{L}_{\text{cam}} + \lambda_{\text{normal}}\,\mathcal{L}_{\text{normal}}. \tag{10}$$

Photometric term. Let $\hat{\mathbf{I}}$ and $\mathbf{I}^*$ denote the rendered and ground-truth images. The photometric term combines a pixel-wise mean-squared error and a perceptual LPIPS loss (Zhang et al., 2018):

$$\mathcal{L}_{\text{photo}} = \lambda_{\text{mse}} \left\|\hat{\mathbf{I}} - \mathbf{I}^*\right\|_2^2 + \lambda_{\text{lpips}}\,\text{LPIPS}(\hat{\mathbf{I}},\, \mathbf{I}^*). \tag{11}$$

Camera term. The camera term is a sum over all ordered view pairs of a Huber loss $\mathcal{L}_{\text{trans}}$ on the relative translation and an angular loss $\mathcal{L}_{\text{rot}}$ on the relative rotation:

$$\mathcal{L}_{\text{cam}} = \omega_t\,\mathcal{L}_{\text{trans}} + \omega_r\,\mathcal{L}_{\text{rot}}. \tag{12}$$

The pairwise form is invariant to the choice of global coordinate frame and provides denser supervision than per-view absolute regression, since every view pair contributes an independent constraint.

Normal term. Let $\mathcal{V}$ be the set of valid pixels (those satisfying the geometry mask, the finite-value check, and an optional object mask). The normal term is a cosine-similarity loss between the refined normal $\mathbf{n}_{\text{ref}}$ and the monocular teacher normal $\mathbf{n}_{\text{tch}}$:

$$\mathcal{L}_{\text{normal}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \big(1 - \mathbf{n}_{\text{ref},i}^{\top} \mathbf{n}_{\text{tch},i}\big). \tag{13}$$

A.3 Training Protocol

Pre-training initialization. The backbone and decoder weights are initialized from PI3 (Wang et al., 2025c), a pretrained pose-free Gaussian splatting model. The normal refinement U-Net and all triangle-specific adapter parameters are initialized from scratch with zero-initialization as described above.

Training schedule. All training and testing are conducted on NVIDIA A100 GPUs. For RE10K, we train for 150K steps at 224×224 resolution. For DL3DV, we first train for 100K steps at 224×224 resolution, then continue training for another 100K steps at 224×448 resolution. The number of context views is sampled uniformly from $[2, 8]$ per iteration during multi-view training, and the context frame gap warms up from $[40, 50]$ to $[50, 200]$ frames on RE10K and from $[15, 30]$ to $[20, 50]$ on DL3DV. Unless otherwise noted, the learning rate is 5×10^{-5} with a backbone multiplier of $0.01\times$ and batch size 1 per GPU. The fixed 6-view checkpoints use the same dataset-specific step budgets and resolution schedules, but keep the number of context views fixed to 6; for this setting, the learning rate is raised to 1×10^{-4} and the batch size is increased to 4 across 4 GPUs. The YoNoSplat baseline follows the same training strategy as our model for controlled comparison.

Resolution and fair comparison. Following YoNoSplat (Ye et al., 2025), we adopt its resolution and fair-comparison protocol for novel-view synthesis: we use the 224×224 version of our model because it best aligns with the experimental settings of the other baselines. Different prior methods adopt different input resolutions: MVSplat (Chen et al., 2024b) and NoPoSplat (Ye et al., 2024) use 256×256, while DepthSplat (Xu et al., 2025a) uses 256×448. Due to computational constraints and to avoid noise from in-house reproduction, it is not feasible to retrain every baseline and our model at a unified resolution. As in YoNoSplat, we keep comparisons fair in two ways. First, our 224×224 model has the smallest receptive size among the compared methods; because all methods first center-crop and then resize their inputs, square crops provide the most conservative receptive coverage. Second, since our model uses the smallest receptive size, rendered outputs from other methods can be center-cropped and resized so that all quantitative and qualitative comparisons are performed on the same image content.

Optimizer. We use AdamW with a linear warm-up of 2,000 steps and gradient clipping at 0.5.

Scheduled sampling. For the pose-free RE10K model, ground-truth camera poses are used with probability $\epsilon(t)$. We set $\epsilon(t) = 1$ before step 80K and decrease it linearly to 0.9 between steps 80K and 100K. Equivalently, the probability of using predicted poses increases from 0 to 0.1 over this interval and remains at 0.1 until the end of the 150K-step training schedule.

Progressive sharpening schedules. During rendering, the opacity temperature $\tau(t)$ increases linearly from $\tau_{\text{init}} = 1.0$ to $\tau_{\text{final}} = 5.0$ over 16,000 steps. The blur multiplier $\beta(t)$ decays from $\beta_{\text{init}} = 1.0$ to $\beta_{\text{final}} = 0.5$ over 16,000 steps.

Large-loss filtering. After 40K warm-up steps, training samples whose total loss exceeds 0.2 (or with MSE > 0.06 or pose loss > 1.0) have their loss contribution scaled to a negligible value.

A.4 Mono-Normal Teacher

The monocular normal teacher is the Omnidata DPT normal estimator (Eftekhar et al., 2021) with a ViT-B/ResNet-50 hybrid backbone (variant "vitb_rn50_384"). Teacher normals are computed offline for every input view and resized to match the prediction resolution via bilinear interpolation. The bootstrap schedule described in Sec. 3.2 of the main paper uses $t_{\text{tk}} = 6{,}000$ steps and $t_{\text{bl}} = 20{,}000$ steps.

B Baseline Mesh Extraction: Quantitative Details

The main paper describes the TSDF fusion pipeline used for Gaussian baselines and the direct export pipeline of TriSplat at a conceptual level. Here we report the exact numerical parameters used in both pipelines, which are necessary for reproducibility.

TSDF fusion parameters. The TSDF volume uses a voxel size of 0.005, SDF truncation of 0.1, and depth truncation of 5.0. Pixels with rendered alpha below 0.3 are masked out. During post-processing,

connected component analysis retains the 50 largest clusters and removes clusters with fewer than 50 triangles.

Direct export parameters. The opacity threshold for triangle pruning is 0.10 after temperature scaling with τ set to the final training temperature of 5.0. Vertex deduplication uses quantized position hashing at precision 10^{-5} with normal-octant keying to prevent merging across opposing face orientations. Per-triangle colors are computed from the 0th-order SH coefficients as $\mathbf{c} = \mathrm{clamp}(\mathrm{SH}_0 \cdot C_0 + 0.5,\ 0,\ 1)$ with $C_0 = \frac{1}{2\sqrt{\pi}} \approx 0.282$. The entire export completes in less than 0.1 s on a single GPU, compared to more than 15 s for TSDF fusion.

C Primitive Rendering Comparison

The main paper reports rendering quality under *mesh rendering*, which matches the simulation-ready objective of TriSplat: a standard triangle rasterizer consumes the exported mesh directly. For completeness, we also report *primitive rendering*, in which each method renders using its own native rasterizer prior to any mesh export (Gaussian splatting for baselines, triangle splatting for TriSplat). Tables A and B report primitive-rendering metrics on DL3DV and RE10K, and Tab. C summarizes the primitive-to-mesh PSNR degradation for each method. Qualitative primitive-rendering comparisons are shown in Fig. A.

Observations. Under primitive rendering, Gaussian baselines attain their strongest numerical scores because their smooth radial falloff provides locally forgiving gradient coverage at render time. When the same models are consumed as meshes, however, the TSDF-fusion step discards these smooth primitives and yields a substantially lower mesh-rendering PSNR (see Tab. C). TriSplat exhibits a markedly smaller primitive-to-mesh degradation because the rendering primitives *are* the exported triangles, so no information is lost during mesh construction. This property is central to the simulation-ready claim: the same representation used during training and inference can be consumed as a mesh with minimal quality loss, without relying on fragile post-hoc surface extraction.

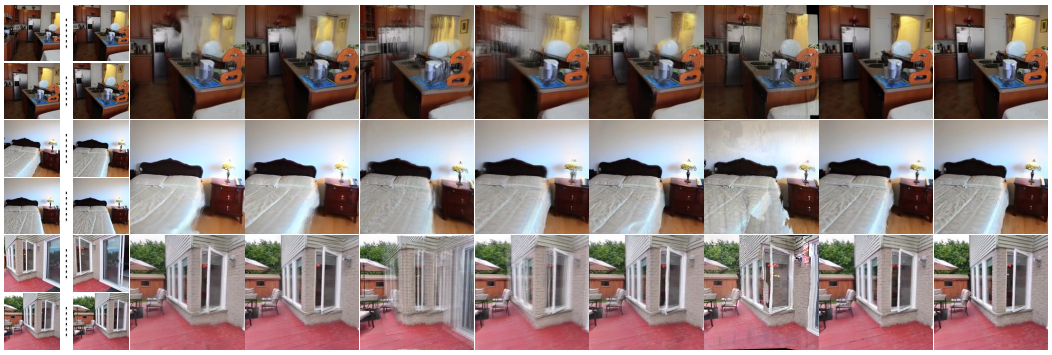


Figure A: **Primitive-rendering comparison on RE10K.** Each method is rendered with its own native rasterizer (Gaussian splatting for baselines, triangle splatting for TriSplat). Compared with the mesh-rendering results in Fig. 5, Gaussian baselines appear closer to TriSplat under primitive rendering, but this quality does not transfer to the exported mesh used by downstream simulation and graphics pipelines.

Table A: **Primitive rendering on DL3DV.** Metrics reported under each method's native rasterizer. Mesh-rendering results under the same evaluation protocol are reported in the main paper (Tab. 2).

Method	6 views			12 views			24 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVSplat	22.83	0.772	0.190	21.29	0.721	0.244	19.93	0.672	0.293
DepthSplat	23.56	0.806	0.150	21.89	0.759	0.196	20.09	0.699	0.256
AnySplat	19.79	0.579	0.257	19.95	0.587	0.272	20.21	0.608	0.274
YoNoSplat	23.88	0.793	0.154	22.91	0.762	0.180	22.22	0.740	0.199
TriSplat (Ours)	23.22	0.768	0.218	21.87	0.726	0.260	21.16	0.704	0.282



Figure B: **Primitive-rendering comparison on DL3DV.** Rows correspond to 6, 12, and 24 input views, and each method is rendered with its own native rasterizer before mesh export. These results provide the counterpart to Fig. 3: Gaussian baselines can look substantially stronger before TSDF fusion, but their native rendering quality is not the representation consumed by standard mesh pipelines.

Table B: **Primitive rendering on RE10K (6 views).** Metrics are reported under each method's native rasterizer before mesh export. These numbers contextualize Tab. 3: Gaussian baselines are competitive under their original splatting renderers, but the downstream mesh-rendering protocol exposes the quality loss introduced by TSDF conversion.

Method	PSNR↑	SSIM↑	LPIPS↓
MVSplat	24.27	0.839	0.146
DepthSplat	25.32	0.849	0.139
AnySplat	17.78	0.551	0.348
YoNoSplat	27.19	0.889	0.106
MeshSplat	22.90	0.788	0.188
SurfelSplat	17.86	0.573	0.373
TriSplat (Ours)	26.46	0.870	0.130

Table C: **Primitive→Mesh degradation on RE10K (6 views).** We report PSNR under primitive rendering (Prim.) and mesh rendering (Mesh), and their difference Δ. A smaller $|\Delta|$ indicates that the exported mesh faithfully preserves the rendering quality of the native primitives.

Method	Prim. PSNR	Mesh PSNR	ΔPSNR
MVSplat	24.27	13.97	−10.30
DepthSplat	25.32	21.23	−4.09
YoNoSplat	27.19	21.94	−5.25
MeshSplat	22.90	19.97	−2.93
SurfelSplat	17.86	11.18	−6.68
TriSplat (Ours)	26.46	24.69	−1.77

D Opacity Temperature Analysis

The opacity transformation applies a scheduled temperature to the predicted opacity.

At $\tau = 1$, the mapping reduces to the identity $o = p$. For $\tau > 1$, values below 0.5 are moved toward zero and values above 0.5 are moved toward one, while the ordering of opacity values is preserved. The midpoint is fixed for every temperature:

$$o(0.5; \tau) = 0.5.$$

In the limiting case,

$$\lim_{\tau \to \infty} o(p; \tau) = \mathbf{1}_{p>0.5},$$

for every $p \in (0, 1)$. In practice, the opacity is clipped to $[10^{-6}, 1 - 10^{-6}]$ before the logit operation for numerical stability. The reported experiments use a linear temperature schedule from $\tau_{\text{init}} = 1.0$ to $\tau_{\text{final}} = 5.0$ over $16{,}000$ steps. An alpha floor of 0.02 is applied during early training to prevent premature pruning of uncertain primitives.

E Triangle Adapter Details

The main paper describes the triangle construction process at the formula level (Eq. (2)). Here we provide code-level details necessary for reproduction.

The canonical equilateral template uses three vertices:

$$(0,\ 0.577,\ 0), \quad (-0.5,\ -0.289,\ 0), \quad (0.5,\ -0.289,\ 0).$$

It is pre-scaled by a factor of 4. The three sigmoid-mapped scale logits are bounded to $[s_{\min},\ s_{\max}]$ and converted to world-space sizes using the predicted depth and a pixel-footprint multiplier derived from the inverse intrinsic matrix. During Stage 1 on RE10K the range is $[0.5, 18.0]$; during Stage 2 it is $[1.2, 15.0]$.

An optional coverage boosting mechanism increases the scale of low-confidence triangles. When the opacity falls below a threshold of 0.20, the scale is boosted proportionally to the gap between the threshold and the opacity, encouraging uncertain triangles to cover a wider area and receive more photometric gradients.

The blur parameter $\hat{\sigma}$ is converted to a positive value via $\sigma = \text{sigmoid}(\hat{\sigma}) \cdot \beta(t) + \epsilon$, where $\beta(t)$ decays linearly from 1.0 to 0.5 over 16,000 steps.

At pixels where the geometry-based tangent-frame rotation is valid, the network's predicted quaternion is overridden by the geometry-derived quaternion. At invalid pixels (boundary pixels and degenerate cross products) the network quaternion is retained as a fallback.

F Additional Results on ScanNet

Tab. D reports novel-view synthesis results on ScanNet under the zero-shot setting, using models trained on RE10K without fine-tuning. Following the convention adopted in the main paper, we report mesh rendering as the primary metric, with primitive rendering included for reference. Despite the significant domain gap between real-estate walkthrough videos and indoor scans, TriSplat maintains competitive performance under mesh rendering. The primitive-to-mesh degradation observed on the training datasets carries over to this unseen domain: Gaussian baselines lose a large margin of PSNR when their TSDF meshes are rendered as triangles, while TriSplat remains stable, confirming that the direct-export property is a domain-robust effect rather than a dataset-specific artifact.

G Additional Ablation Studies

The main-paper ablation (Tab. 5) validates the four key design choices at a coarse level. Here we provide finer-grained studies on hyperparameters and architectural variants: the triangle scale range in Appendix G.1, the blur schedule in Appendix G.2, and the opacity temperature schedule

Table D: **Novel-view synthesis on ScanNet (zero-shot, 6 views).** Models trained on RE10K, evaluated without fine-tuning. Primitive rendering (Prim.) is each method's native rasterization; mesh rendering (Mesh) rasterizes the exported mesh.

Method	Prim.			Mesh		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVSplat	17.64	0.639	0.345	11.72	0.320	0.625
DepthSplat	17.15	0.589	0.438	16.06	0.534	0.519
AnySplat	13.18	0.420	0.571	12.67	0.430	0.598
YoNoSplat	22.99	0.809	0.240	16.69	0.587	0.445
MeshSplat	17.62	0.633	0.352	14.35	0.482	0.578
SurfelSplat	12.69	0.329	0.592	11.05	0.385	0.752
TriSplat (Ours)	22.61	0.796	0.265	17.03	0.540	0.530

in Appendix G.3. Consistent with the main paper, all tables in this section report surface geometry (CD, Precision) together with mesh-rendering quality (PSNR, LPIPS), so that every design choice is evaluated on the same simulation-ready metrics used in the main experiments.

G.1 Triangle Scale Range

Tab. E examines the effect of the triangle scale range $[s_{\min}, s_{\max}]$. A range that is too narrow limits the model's ability to cover large surface regions, reducing recall. A range that is too wide permits excessively large triangles that introduce rendering artifacts.

Table E: **Ablation on triangle scale range** (RE10K, 6 views).

$s_{\min}$	$s_{\max}$	CD↓	Prec.↑	PSNR↑	LPIPS↓
0.5	10.0	0.189	0.619	22.14	0.375
0.5	18.0	0.190	0.708	24.69	0.269
1.2	15.0	0.192	0.615	22.49	0.357
1.2	25.0	0.181	0.631	19.97	0.452

G.2 Blur Schedule

Tab. F isolates the effect of blur scheduling. Without scheduling (fixed low blur) early training suffers from poor gradient coverage. A fixed high blur allows stable training but produces soft surfaces. The default schedule decaying from 1.0 to 0.5 over 16K steps achieves the best balance.

Table F: **Ablation on blur scheduling** (RE10K, 6 views).

Blur strategy	CD↓	Prec.↑	PSNR↑	LPIPS↓
Fixed low ($\beta = 0.1$)	0.185	0.640	21.44	0.402
Fixed high ($\beta = 1.0$)	0.223	0.542	20.90	0.434
Schedule $1.0 \to 0.5$ (8K)	0.217	0.571	21.05	0.418
Schedule $1.0 \to 0.5$ (16K)	0.190	0.708	24.69	0.269
Schedule $1.0 \to 0.1$ (16K)	0.212	0.588	21.26	0.384

G.3 Opacity Temperature

Tab. G studies the opacity temperature schedule. Without temperature scaling ($\tau = 1.0$ throughout) the opacity distribution remains soft and the resulting semi-transparent surfaces degrade mesh quality. A very high final temperature ($\tau = 25.0$) produces near-binary opacities that cause gradient instability.

H Additional Simulation Experiments

The main paper summarizes robotic grasping, ball dynamics, and multi-platform locomotion in Unity and NVIDIA Isaac Sim (Fig. 9). Here we expand the simulation demonstrations into four-frame dynamic sequences using the directly exported triangle meshes without any manual cleanup or format conversion. Frames are ordered from left to right by time. Rigid-body dynamics are reported in Appendix H.1 and legged locomotion in Appendix H.2.

Table G: **Ablation on opacity temperature** (RE10K, 6 views).

Temperature schedule	CD↓	Prec.↑	PSNR↑	LPIPS↓
Fixed $\tau = 1.0$	0.183	0.637	20.88	0.439
τ: $1.0 \rightarrow 5.0$ (16K)	0.190	0.708	24.69	0.269
τ: $1.0 \rightarrow 10.0$ (16K)	0.204	0.560	21.87	0.379
τ: $1.0 \rightarrow 25.0$ (16K)	0.232	0.454	17.23	0.477

H.1 Rigid-Body Dynamics

We evaluate the physical utility of our reconstructed meshes through two rigid-body scenarios simulated in NVIDIA Isaac Sim with the PhysX backend. First, in a ball drop experiment, a sphere is released from various heights onto the surface. The resulting collision responses–including intricate bounce trajectories–demonstrate that our mesh faithfully captures the underlying geometry with high fidelity. Second, in an object stacking experiment, we place multiple rigid objects on the reconstructed surfaces. The sustained stability of these stacks highlights the exceptional surface flatness and normal consistency achieved by our method, ensuring reliable contact physics for downstream interaction tasks.

H.2 Legged Locomotion

A simulated quadruped robot traverses outdoor scenes reconstructed by TriSplat. The robot's foothold planning directly leverages the mesh surface normals and collision geometry. In scenes containing stairs and chairs, the robot successfully navigates the terrain.



Figure C: **Unity character navigation.** The exported TriSplat mesh is imported into Unity as static scene geometry for character navigation and collision handling. The four frames show temporal progression from $t = 1$ to $t = 4$ and demonstrate usable collision surfaces without manual cleanup, conversion from Gaussian primitives, or scene-specific mesh repair.



Figure D: **Unity object interaction.** We load the reconstructed mesh into Unity and interact with the scene using standard engine collision and physics components. The four frames show temporal progression from $t = 1$ to $t = 4$; stable contact with reconstructed surfaces indicates that the exported triangle mesh is directly usable for interactive applications, rather than only for image rendering.

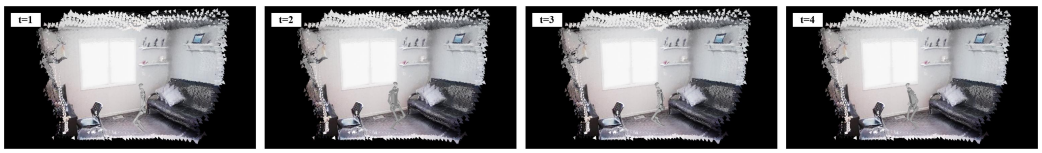


Figure E: **Isaac Sim humanoid locomotion.** The four frames show the H1 humanoid moving over the same exported TriSplat mesh from $t = 1$ to $t = 4$. This sequence expands the compact main-paper demonstration by showing continuous contact-rich locomotion on the reconstructed triangle geometry, without an intermediate reconstruction or mesh-repair stage.

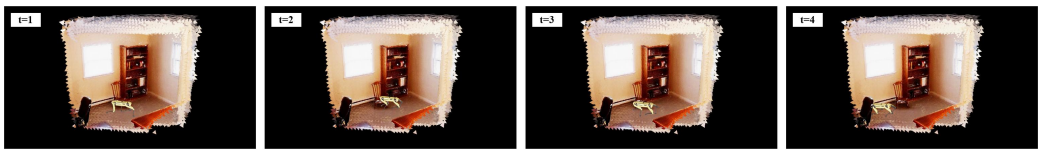


Figure F: **Isaac Sim quadruped locomotion.** A quadruped traverses the exported TriSplat mesh using the simulator's native collision and contact solver. The four frames show temporal progression from $t = 1$ to $t = 4$ and illustrate that the same mesh representation supports different embodied agents and locomotion patterns.

I More Visual Comparisons

We provide additional qualitative comparisons on RE10K (Zhou et al., 2018), DL3DV (Ling et al., 2024), and ScanNet (Dai et al., 2017). These figures exclude scenes already shown in the main-paper visual pages. Fig. G groups the remaining RE10K mesh-rendering comparisons under one caption, Fig. H shows additional DL3DV mesh-rendering examples, and Figs. I and J show additional primitive-rendering examples. Figs. K and L show remaining textured-mesh and depth/normal examples, and Figs. M–N show zero-shot ScanNet rendering results.



Figure G: **Additional mesh-rendering comparisons on RE10K.** The top block is the first RE10K scene excluded from the main-paper Fig. 5 (the main figure shows the remaining two scenes), followed by the supplementary examples. We group all remaining RE10K mesh-rendering examples under one figure to avoid splitting identical comparisons across separate captions. Each group contains six input views, mesh-rendered baseline results, TriSplat, and the ground-truth target view. Since every image is rendered from the exported mesh, artifacts reflect the quality of the representation available to downstream graphics and simulation systems.

J Mesh Evaluation Protocol

The mesh metrics reported in the main paper (Tables 1 and 3) follow the standard protocol of MeshSplat (Chang et al., 2025) and ScanNet (Dai et al., 2017). Both the predicted and ground-truth meshes are sampled into point clouds and voxel-downsampled at resolution 0.02 to ensure uniform density. One-sided distances are computed as $d(A, B) = \frac{1}{|A|} \sum_{\mathbf{a} \in A} \min_{\mathbf{b} \in B} \|\mathbf{a} - \mathbf{b}\|$, yielding Chamfer Distance CD $= d(\mathcal{M}_p, \mathcal{M}_g) + d(\mathcal{M}_g, \mathcal{M}_p)$. Precision and Recall are the fractions of



Figure H: **Additional mesh-rendering comparisons on DL3DV.** These examples are not used in the main-paper visual page. The comparison again isolates mesh-rendering quality: all methods are evaluated after export, so sharper results indicate a mesh that better preserves the original image evidence.



Figure I: **Additional primitive-rendering comparisons on RE10K.** Each method is rendered under its own native rasterizer before mesh export. These examples complement Fig. G by showing that strong Gaussian primitive renderings do not necessarily translate into high-quality exported meshes.



Figure J: **Additional primitive-rendering comparisons on DL3DV.** Native primitive renderings are shown for the same evaluation style as the DL3DV mesh-rendering figures. The contrast with Fig. H illustrates why mesh rendering is the relevant protocol for simulation-ready reconstruction.

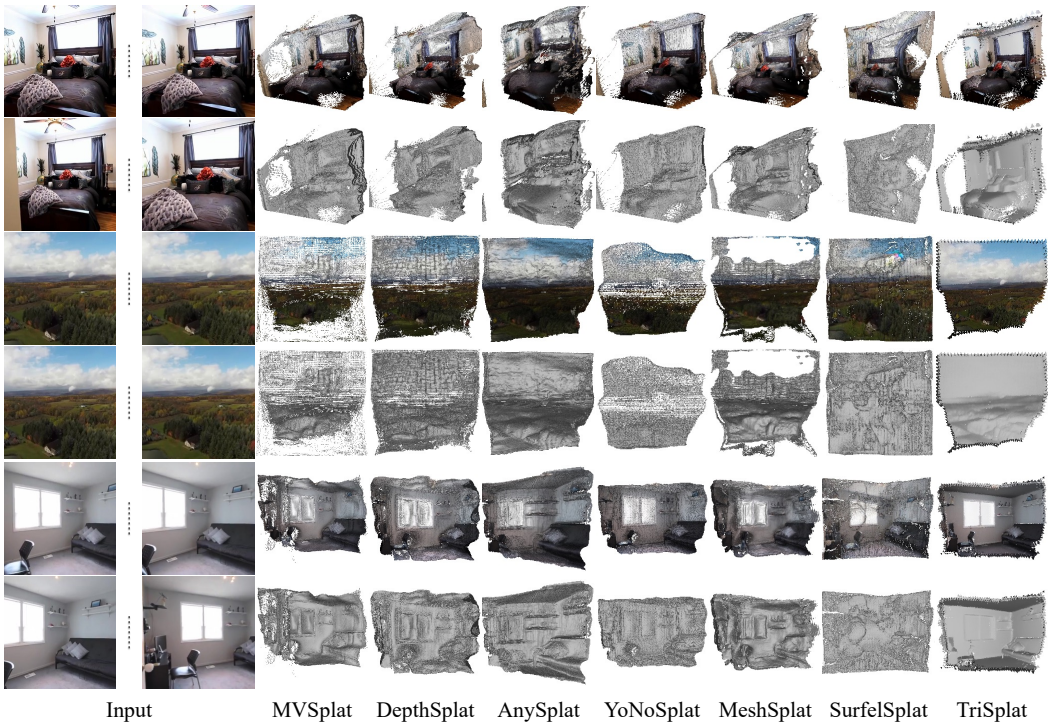


Figure K: **Additional textured mesh comparisons on RE10K.** The top block is the first RE10K textured-mesh scene excluded from the main-paper Fig. 6, followed by the supplementary exported textured meshes. The figure visualizes exported textured meshes rather than target-view renders. TSDF-fused Gaussian meshes often show over-smoothed surfaces, missing thin structures, and fragmented regions, while TriSplat preserves direct triangle geometry and appearance in the exported representation.

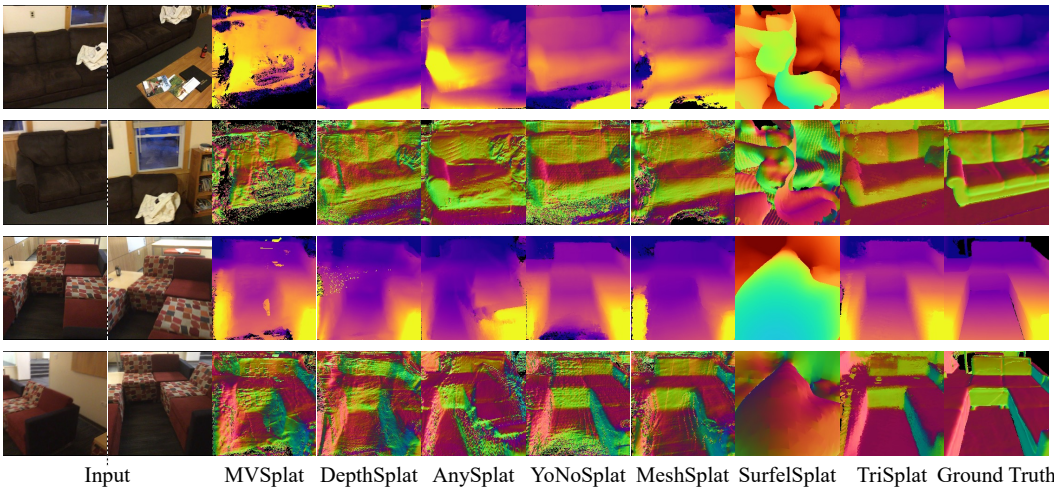


Figure L: **Additional depth and normal comparisons on ScanNet.** The top block is the first ScanNet scene excluded from the main-paper depth/normal figure (Fig. 7), followed by the supplementary example. All methods are trained on RE10K and evaluated on ScanNet without fine-tuning. TriSplat produces smoother, geometrically coherent normals aligned with surface boundaries, while Gaussian baselines often produce noisy orientation fields under the domain shift.



Figure M: **Primitive-rendering comparison on ScanNet.** Each method is rendered with its native primitive representation on zero-shot ScanNet scenes. These fixed-resolution renders are inserted without border cropping, preserving each method's original output frame and avoiding artificial enlargement or truncation of any baseline result.

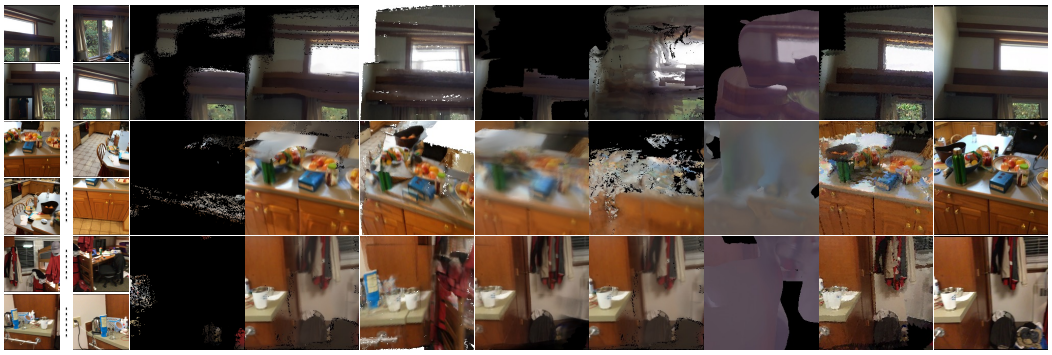


Figure N: **Mesh-rendering comparison on ScanNet.** We render each exported mesh with the same triangle pipeline in the zero-shot setting. Fixed-resolution outputs are inserted without border cropping, so the comparison preserves the full rendered frame for every baseline, including methods whose exported meshes contain large empty regions or incomplete surfaces.

predicted and ground-truth points within distance $\delta = 0.05$ of each other, and the F1 score is their harmonic mean. All nearest-neighbor queries use KD-trees (Open3D) and all metrics are computed in the world coordinate frame.