LOOMMESH: FEED-FORWARD MESHSPLATTING

Anonymous authors
Paper under double-blind review



Figure 1: **LoomMesh reconstructs a textured triangle mesh from two posed images in a single forward pass.** Top: the two input views of four RealEstate10K test scenes. Bottom: the mesh predicted for each pair, whose cross-view faces join the vertices of both views, rendered from a novel viewpoint exactly as exported, with no per-scene optimisation or post-processing.

ABSTRACT

3D Gaussian Splatting methods enable efficient optimisation and rendering, but are not natively supported by widely used, highly optimised mesh-based graphics engines (*e.g.*, Unity, Blender). To exploit these engines, mesh-based neural rendering with *triangle* primitives has recently emerged, yet it typically relies on per-scene optimisation over many posed views, making it slow and hard to scale. We study ***feed-forward mesh splatting***: creating a mesh directly from a single pair of posed images in one forward pass. This is challenging due to (1) sparse observations and (2) the need to jointly infer vertex positions (*i.e.*, geometry) and their connectivity (*i.e.*, surface). To this end, we propose **LoomMesh**, which unprojects every pixel of each view to a 3D vertex carrying geometry and appearance features. We then connect the vertices in a candidate graph, linking each vertex to its same-view neighbours by pixel location and to its cross-view neighbours by geometric reprojection. To learn vertex geometry and connectivity jointly, we introduce a topology reasoning operator that, shared across all vertices, refines each vertex's depth and decides which candidate faces to keep, passing evidence between neighbours along surfaces but not across their boundaries. The resulting mesh is rendered differentiably, so LoomMesh is trained end to end and needs no ground-truth meshes. Extensive experiments show that LoomMesh outperforms per-scene optimised methods in both visual fidelity and inference cost, as well as alternatives that combine feed-forward reconstruction with mesh conversion.

1 INTRODUCTION

Mesh reconstruction is an important line of work in 3D vision, as graphics engines and simulators are built on triangle meshes. Recent Gaussian-based splatting methods enable efficient optimisation and rendering and feed-forward variants further reconstruct unseen scenes directly from sparse input views (Yu et al., 2021; Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025). However these

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.

methods focus on rendering quality and obtain a mesh only by post-hoc extraction. To exploit graphics engines directly, triangle splatting (Held et al., 2026b) represents scenes with triangle primitives, and mesh-based approaches such as MeshSplatting (Held et al., 2026a) further incorporate connectivity to produce explicit meshes ready for these engines. Nevertheless, these methods typically rely on per-scene optimisation over many posed views, making reconstruction slow and hard to scale.

To go a step further, we study *feed-forward mesh splatting*: reconstructing an engine-ready explicit mesh directly from a single pair of posed images in one forward pass. The challenge is not simply to replace each Gaussian with a triangle, but to determine how vertices should connect to form a coherent surface, as *Delaunay triangulation* from MeshSplatting cannot be applied in a feed-forward setting. Crucially, vertex positions and connectivity are mutually dependent: depths are refined through the faces that join them. As uncertain vertex positions can suggest spurious connections, which in turn mix evidence across surface discontinuities and further corrupt the geometry. A circularity stalls end-to-end training when we infer one from a moving estimate of the other. Vertex geometry and connectivity must therefore be resolved jointly rather than sequentially.

To this end, we propose **LoomMesh**, the first feed-forward mesh splatting method, which efficiently converts two views into a mesh end to end. Every pixel becomes a vertex, and a candidate graph links it to its image neighbours and, by reprojection, to the other view; its triangles are the candidate faces. Each edge carries a conductance that compares its coplanarity residual with a tolerance shifted by a learned vote, and a face is as opaque as its weakest edge. A topology reasoning operator then updates all vertices in parallel: each reads the matching posterior around its current depth, which keeps the matcher learning, and gathers its neighbours' depth beliefs weighted by conductance, so evidence spreads along surfaces but not across boundaries. Its updated state jointly predicts vertex positions and connectivity, the latter also determine face opacities to link geometry refinement to mesh formation. Shared across all units, the operator is highly parameter-efficient. LoomMesh is trained end to end on images only, requiring neither ground-truth meshes nor surface extraction.

Our *contributions* are: **(1)** a novel, more promising 3D reasoning problem, feed-forward mesh splatting, that reconstruct an explicit mesh from a pair of posed images trained end to end, without per-scene optimisation or mesh conversion; **(2)** the very first solution, LoomMesh, characterised by a shared topology reasoning operator that processes jointly estimate vertex geometry and connectivity; **(3)** extensive experiments demonstrating that LoomMesh outperforms both per-scene optimised mesh splatting methods and alternatives with feed-forward reconstruction and mesh conversion.

2 Related Work

Differentiable Splatting-based Scene Representation. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) represents a scene with explicit 3D Gaussians and renders them by rasterization, which is much more efficient than the per-ray volume rendering of neural radiance fields (Mildenhall et al., 2020). Being volumetric, however, 3D Gaussians do not represent surfaces accurately. 2DGS (Huang et al., 2024) addresses this by flattening them into oriented 2D Gaussian disks aligned with the surface, improving geometric accuracy. Triangle Splatting (Held et al., 2026b) goes further and replaces them with triangles, the standard primitive of computer graphics, rendered as differentiable splats; 2DTS (Sheng et al., 2025) anneals each triangle's window until it is fully opaque. MeshSplatting (Held et al., 2026a) enforces connectivity through a restricted Delaunay triangulation, producing meshes that standard graphics engines render directly. All of these methods rely on time-consuming per-scene optimization over dozens to hundreds of posed images, and degrade sharply when only a few views are available (Table 1).

Generalizable Feed-forward Networks Generalizable feed-forward methods seek to train neural networks on large scale dataset to learn priors across scenes and reconstruct from only a few posed images in a single forward pass. Early methods predict radiance fields conditioned on image features (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021) but inherit the cost of per-ray volume rendering. pixelSplat (Charatan et al., 2024) instead predicts per-pixel 3D Gaussians from an image pair, and later methods place them more accurately with a multi-view cost volume (Chen et al., 2024) and monocular depth priors (Xu et al., 2025), scale up with large transformers (Zhang et al., 2024), or drop the need for input poses (Ye et al., 2025). These methods are limited to novel-view synthesis, but their Gaussians do not form a surface. MeshSplat (Chang et al., 2026) targets surfaces

by predicting 2D Gaussians with cross-view and normal supervision, but still obtains its mesh by TSDF fusion (Curless & Levoy, 1996) of rendered depth, a separate step whose output is not what the network rendered during training. LoomMesh builds on the depth backbone of DepthSplat but predicts the mesh itself: its vertices, their colours and their connectivity.

Mesh connectivity. Connectivity is usually fixed by a rule or decided per scene. Range-image meshing joins neighbouring depth samples closer than a multiple of the sampling resolution (Turk & Levoy, 1994). Delaunay triangulation (Delaunay, 1934) connects points by proximity alone, leaving the choice of faces to graph cuts (Labatut et al., 2007) or per-scene optimisation (Son et al., 2024; Guédon et al., 2025; Held et al., 2026a). None of these learns across scenes which vertices to join. The weak membrane (Blake & Zisserman, 1987) decides continuity jointly with depth, breaking the surface wherever stretching it across a gap costs more than a fixed penalty. LoomMesh uses such a rule as its geometric gate but learns across scenes when to override it, in one forward pass and while refining the depth the rule reads.

3 METHOD

Our goal is to reconstruct an explicit triangle mesh from posed images in a feed-forward manner, without iterative per-scene optimisation or a separate surface-extraction stage. Figure 2 provides an overview. Given N input images $\mathbf{I}^i \in \mathbb{R}^{H\times W\times 3}$ and their camera matrices $\mathbf{P}^i = \mathbf{K}^i[\mathbf{R}^i \,|\, \mathbf{t}^i]$, we learn the mapping

$$f_{\boldsymbol{\theta}} :\ \{(\mathbf{I}^i, \mathbf{P}^i)\}_{i=1}^{N} \ \longmapsto\ \mathcal{M} = \Big(\underbrace{\{(\mathbf{v}_j, \mathbf{c}_j)\}_{j=1}^{NHW}}_{\text{vertices}},\ \underbrace{\mathcal{F},\ \{\alpha_f\}_{f\in\mathcal{F}}}_{\text{faces and opacities}}\Big), \tag{1}$$

where we use $N = 2$ views. Each vertex has a world-space position $\mathbf{v}_j \in \mathbb{R}^3$ and spherical-harmonic coefficients $\mathbf{c}_j$ for view-dependent colour. The candidate face set $\mathcal{F}$ contains two triangles per image-lattice quad and additional faces formed through cross-view reprojection. A face's opacity is determined by its three edge conductances:

$$\alpha_f = \min_{e\in\mathcal{E}(f)} k_e, \qquad k_e \in [0, 1], \tag{2}$$

where $\mathcal{E}(f)$ denotes the edges of face f. These continuous opacities make face selection differentiable during training; at inference, we threshold them to obtain the final mesh.

To predict $\mathcal{M}$, our pipeline is as follows: Section 3.1 lifts every pixel to a vertex in the 3D space using the depth and matching posterior of a multi-view backbone, Section 3.2 connects the vertices in a candidate graph, and Section 3.3 introduces the topology reasoning operator, which refines vertex depths and predicts edge conductances. The vertices and predicted faces are assembled into a mesh, rendered from novel views. Sections 3.4 and 3.5 describe the renderer and the training losses.

3.1 FROM IMAGES TO VERTEX PRIMITIVES

We adopt the backbone of DepthSplat (Xu et al., 2025), which combines multi-view matching with monocular features to predict initial depth at $1/s$ of the input resolution. Given matching logits $h(x)$ over inverse-depth hypotheses d_k, the posterior and initial inverse depth are

$$p(k \mid x) = \mathrm{softmax}_k\, h(x), \qquad \hat{d}^{\,0}(x) = \sum_k p(k \mid x)\, d_k. \tag{3}$$

The initial depth, log-posterior $\log p$, and monocular features are passed to our topology reasoning operator, which jointly refines vertex depths and connectivity.

After refinement, a DPT head (Ranftl et al., 2021) upsamples the coarse depth to full resolution. We then unproject each pixel into a world-space vertex:

$$\mathbf{v}_x = \mathbf{R}^{i\top}\big(z(x)(\mathbf{K}^i)^{-1}\tilde{x} - \mathbf{t}^i\big), \tag{4}$$

where $\tilde{x}$ is the homogeneous pixel coordinate in view i and $z(x)$ is its refined depth. This produces one vertex per pixel per view. A second DPT head predicts the per-vertex spherical-harmonic coefficients $\mathbf{c}_j$, completing the vertex geometry and appearance in equation 1.

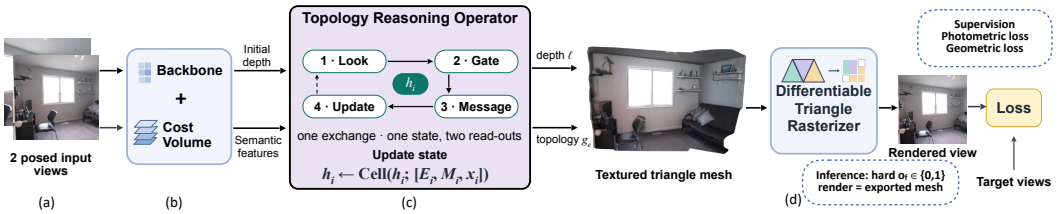

Figure 2: **Overview of LoomMesh.** Given two posed input views (a), a backbone with a multi-view cost volume (b) produces initial depth estimates and semantic features. The depths are lifted into pixel-aligned vertices, with candidate connections formed from image neighbours and cross-view reprojection. The topology reasoning operator (c) reads matching evidence at the current depths, gates messages between neighbouring vertices, and updates their states to refine depth and predict connectivity. The resulting vertices and faces form a textured triangle mesh, which is rendered by a differentiable triangle rasteriser (d) for end-to-end training with photometric and geometric losses.

3.2 Candidate Graph and Topology Gate

Candidate graph. To restrict the hypothesis space of possible meshes we want to exploit the geometric priors already exist in our input views. In geometric deep learning, images are usually treated as grid graphs (Bronstein et al., 2021), which encode the basic inductive bias of a pixel-aligned representation: adjacent pixels of the same view usually sample the same surface, so image pixel adjacency is the default connectivity for the 3D space.

Given the vertices of equation 4, we enumerate a candidate edge set $\mathcal{E}$ every forward pass from two families. *Grid links* join each vertex to its horizontal and vertical 4-neighbours within a view, plus one consistently oriented diagonal per pixel quad, so that every quad is exactly two triangles. *Cross-view links* are the only route between views: every vertex of view A is projected through the camera of view B at its current depth and linked to the pixel it lands in and that pixel's 4-neighbours, in both directions, keeping the *eight* shortest links per vertex. Candidate faces $\mathcal{F}$ are the 3-cliques of $\mathcal{E}$.

Geometric evidence. Whether an edge should exist is first a geometric question: An edge should join two vertices on one surface, but not across an occlusion boundary. Edge length cannot tell a slanted surface from a depth jump, so we test coplanarity with the point-to-plane distance (Chen & Medioni, 1992). With $\mathbf{n}_i$ the normal of vertex i from its lattice neighbours which defines its tangent plane, and $\Delta_e = \mathbf{v}_j - \mathbf{v}_i$, edge $e = (i, j)$ has residual ρ_e and geometric evidence η_e,

$$\rho_e = \max\left(|\mathbf{n}_i \cdot \Delta_e|,\, |\mathbf{n}_j \cdot \Delta_e|\right) / \bar{\phi}_e, \qquad \eta_e = \log \rho_\star - \log \rho_e, \tag{5}$$

The residual stays near zero on a surface at any slant and grows with the gap across a boundary. Its normaliser $\bar{\phi}_e$ is the smallest depth difference the cost volume can resolve at the two endpoints (about half a bin; App. B.3), so ρ_e counts resolvable steps at every depth. The evidence compares it with a fixed tolerance $\rho_\star = 4$.

Edge conductance. We use the term *conductance* to define our edge connectivity, borrowing it from anisotropic diffusion (Perona & Malik, 1990): every candidate edge carries a conductance

$$k_e = \sigma\big(\eta_e - g_e\big), \tag{6}$$

a soft comparison of the geometric evidence against a learned vote g_e that carries the appearance and accumulated evidence the residual cannot see. With η_e fixed, the vote g_e is the only way the network can change a conductance; it is zero at initialisation, so the system starts as pure geometry, and it is produced by the operator of section 3.3.

3.3 Topology Reasoning Operator

Motivation. Trained end to end, feed-forward Gaussian models keep refining their depth, because a Gaussian renders acceptably wherever its depth places it. A key challenge to our mesh model is that depth and connectivity depends on each other as depth is trained from images rendered with those faces, a circular dependency. Left alone, early depth is smooth across boundary, so faces stretch over it and photometric loss can neither cut such a face. Topology improvement stalls early

and the matching posterior saturates. Therefore, we introduce a *topology reasoning operator* that addresses both sides at once. In one step it refines each vertex's depth and predicts the conductance of its edges.

Gated graph exchange and update. To decide depth and connectivity together, the operator gives each vertex i a single state $\mathbf{h}_i$ from which both are read out (equation 10). The state gathers three kinds of evidence: the vertex's own matching evidence $\mathbf{E}_i$, messages $\mathbf{M}_i$ from vertex's neighbours, and its own inputs $\mathbf{x}_i$. A single exchange updates the state from all three with a gated update cell, a standard GRU cell (Cho et al., 2014) as used for node updates in gated graph networks (Li et al., 2016):

$$\mathbf{h}_i' = \mathrm{Cell}\big(\mathbf{h}_i,\, [\mathbf{E}_i, \mathbf{M}_i, \mathbf{x}_i]\big). \tag{7}$$

Its update gate lets a vertex with decisive evidence keep its state and one with flat evidence defer to its neighbours. The operator runs on the $(H/s) \times (W/s)$ grid of the posterior, before the DPT upsampler, where it reads the posterior in place and its candidate graph (section 3.2) is s^2 smaller.

Node and edge inputs. Each vertex enters with $\mathbf{x}_i = [\ell_i,\, c_i,\, \mathbf{Pf}_i]$: its log inverse depth $\ell_i = \log \hat{d}_i^0$ (mean-centred), its confidence $c_i = \max_k p(k \mid x)$, and a 32-channel projection of the DINOv2 ViT descriptor $\mathbf{f}_i$ from the monocular branch. The initial state is $\mathbf{h}_i^{(0)} = \mathbf{W}_{\mathrm{in}}\mathbf{x}_i \in \mathbb{R}^{64}$, and each edge carries an embedding: $\mathbf{e}_{ij} = \mathbf{W}_{\mathrm{e}}[\mathbf{1}_{\mathrm{fam}(ij)}, \log \rho_{ij}, \log \lambda_{ij}]$ which gives us its edge family, plane residual (equation 5) and length λ_{ij} in resolvable units at the initial depth.

Gated message passing. A vertex's neighbours carry evidence about the surface it lies on, so vertices exchange messages over the coarse graph, where one hop spans s pixels or reaches the other view. Each edge reads the vote of its gate (equation 6) from the states at both ends, and each vertex averages its neighbours' messages with conductance weights:

$$g_e = \varphi_g\big(\mathbf{h}_i + \mathbf{h}_j,\, |\mathbf{h}_i - \mathbf{h}_j|,\, \mathbf{e}_{ij}\big), \qquad \mathbf{M}_i = \frac{\sum_{j\in\mathcal{N}(i)} k_{ij}\, \varphi_m\big(\mathbf{h}_j, \mathbf{e}_{ij}, \ell_j - \ell_i, c_j\big)}{\sum_{j\in\mathcal{N}(i)} k_{ij}}, \tag{8}$$

with two-layer MLPs φ_g, φ_m and $k_{ij} = k_e$; the sum and absolute difference make the vote symmetric. Messages carry the sender's depth relative to the receiver, and a neighbour across a likely boundary gets almost no weight, so evidence spreads along a surface but not across its boundary, as in anisotropic diffusion (Perona & Malik, 1990) and gated GCNs (Bresson & Laurent, 2017).

Evidence at the current estimate. To keep the matcher learning, each vertex also reads its log-posterior around its current estimate, like the correlation lookup of RAFT (Teed & Deng, 2020):

$$\mathbf{E}_i = \big[\log p_i(b_i + o)\big]_{o=-w}^{w}, \qquad b_i = (D-1)\,\big(e^{\ell_i} - d_{\min}\big)/\big(d_{\max} - d_{\min}\big), \tag{9}$$

with $w = 2$ and linear interpolation. The read is relative: it shows a vertex how well its depth is supported and whether a step nearer or farther does better, for the depth step to act on. It also keeps the matcher learning: unlike the soft-argmax, it still passes gradient to the matching scores once the posterior is confident (Appendix B.5).

Two read-outs. The updated state is read twice: at each coarse vertex for a bounded depth step, and, upsampled bilinearly to $\hat{\mathbf{h}}$, on each full-resolution edge for a face vote:

$$\ell_i \leftarrow \ell_i + \delta_{\max} \tanh \varphi_z(\mathbf{h}_i^{(1)}), \qquad g_e^{\mathrm{face}} = \varphi_{\mathrm{face}}\big(\hat{\mathbf{h}}_i + \hat{\mathbf{h}}_j,\, |\hat{\mathbf{h}}_i - \hat{\mathbf{h}}_j|,\, \mathbf{1}_{\mathrm{fam}(e)},\, \log \rho_e\big). \tag{10}$$

With $\delta_{\max} = 0.05$ (about 5% in depth), the step refines the matching estimate rather than overriding it, and the rendering loss reaches the lookup through the DPT upsampler, φ_z and the update cell. The face vote and enters the same gate, $k_e = \sigma(\eta_e - g_e^{\mathrm{face}})$, where a face takes the conductance of its weakest edge, $\alpha_f = \min_{e\in f} k_e$. Both heads output zero at initialisation, so training starts from the matching depth and the geometric gate alone (details in Appendices B.3 and B.4).

3.4 Differentiable triangle ray-splatting

The renderer turns the vertices, their SH colours and the face opacities α_f into colour, depth and coverage, all differentiable. Like 2DTS (Sheng et al., 2025), it intersects rays with the triangles analytically, so hits are ordered exactly per ray rather than by a screen-space sort. 2DTS and Triangle

Splatting (Held et al., 2026b) also soften each triangle with a window that decays *inside* it, and 2DTS and MeshSplatting annealed to a hard face over a per-scene schedule. We keep the closed triangle exact and add a soft band *outside* it, no annealing is needed, and α_f is purely the gate's existence probability, not a coverage relaxation.

Rendering model. Each pixel casts a ray $\mathbf{o} + t\,\mathbf{d}$ and intersects the faces by Möller–Trumbore (Möller & Trumbore, 1997), which gives the hit distance t and the barycentrics λ, kept even outside the triangle, where some $\lambda_m < 0$. With $d_m = \lambda_m h_m \bar{f}/t$ the signed distance in pixels from the hit to the edge opposite vertex m (h_m its altitude, t the hit distance, $\bar{f}$ the focal length in pixels; negative outside the triangle), a hit's opacity is the face opacity times an exterior window,

$$a_i = \alpha_{f_i} I_{f_i}, \qquad I_f = \prod_{m=1}^{3} \operatorname{clamp}\big(1 + d_m/\sigma,\ 0,\ 1\big), \tag{11}$$

so $I_f = 1$ on the closed triangle and falls linearly to 0 within $\sigma = 0.7\,\text{px}$ outside it. The $k = 8$ nearest hits, in exact t order, are composited into colour C, depth D and coverage A, with $T_i = \prod_{j<i}(1 - a_j)$:

$$C = \sum_i T_i a_i\, \mathbf{c}_i + T_{k+1}\, \mathbf{c}_{\text{bg}}, \qquad D = \sum_i T_i a_i\, t_i, \qquad A = 1 - T_{k+1}, \tag{12}$$

with $\mathbf{c}_i$ the barycentric interpolation of the vertices' SH colours in direction $\mathbf{d}$.

Gradients. Opacities receive the standard compositing gradient (Kerbl et al., 2023), so colour, depth and coverage errors all reach the topology gate through α_f. Vertices receive the analytic Möller–Trumbore Jacobian of (t, λ). Since I_f is flat on the closed triangle, interior hits are differentiated exactly as by a hard rasterizer; only hits in the band pull the two vertices of the nearby edge, which gives the coverage gradient that hard rasterization lacks.

Implementation and mesh export. In the feed-forward setting the scene changes at every step, faces are binned to 16×16 tiles as in 3DGS (Kerbl et al., 2023) rather than stored in a BVH, and each pixel keeps its k nearest hits by exact t (Appendix B.6). At evaluation, α_f is binarised at 0.5 and $\sigma = 0$, so equation 12 reduces to first-hit shading of the surviving faces. With their shared vertices and colours, these faces are the exported mesh, connected and without post-processing: what is trained is what is exported.

3.5 Training objectives.

LoomMesh is trained end-to-end from posed images; no depth, normal or topology labels are used. The objective is

$$\mathcal{L} = \mathcal{L}_{\text{mse}} + \lambda_{\text{lp}}\mathcal{L}_{\text{lpips}} + \lambda_{\text{n}}\mathcal{L}_{\text{normal}} + \lambda_{\text{gm}}\mathcal{L}_{\text{gm}} + \lambda_{\text{w}}\mathcal{L}_{\text{warp}}. \tag{13}$$

The photometric terms, an L_2 loss and LPIPS (Zhang et al., 2018) on the rendered target views, are the primary signal: through the renderer they reach the colours, the vertices and the face opacities, so photometric error alone can delete a face. Two geometric terms compare the rendered views with the target views: $\mathcal{L}_{\text{normal}}$, a cosine loss on the rendered normals, and $\mathcal{L}_{\text{gm}}$, the multi-scale gradient-matching loss of MiDaS (Ranftl et al., 2022) on the log of the expected depth D/A. $\mathcal{L}_{\text{gm}}$ is blind to global scale and responds mainly to edges the mesh has missed or invented, so absolute depth stays with the cost volume and the photometric loss. $\mathcal{L}_{\text{warp}}$ warps each context depth map into the other view and penalises its log-depth gap to the prediction there (Bian et al., 2019), since photometric supervision alone permits doubled surfaces. More details of the losses are in section A.

4 Experiments

4.1 Experimental Setup

Implementation. We implement LoomMesh in PyTorch, adopting the depth backbone of DepthSplat (Xu et al., 2025) with Depth Anything V2 and DINOv2. We train for 100k iterations on four

Table 1: Comparison on RealEstate10K (256^2). Reported results evaluate meshes. Best available results are in **bold**;

			Novel-view synthesis			Geometry Quality		Mesh size (K)	
Method	Reconstruction	Iterations	PSNR↑	SSIM↑	LPIPS↓	L1↓	Cos.↑	Vertices↓	Faces↓
Gaussian-based methods									
3DGS	Per-scene opt.	30k	14.97	0.524	0.487	0.540	0.302	3113	5837
2DGS	Per-scene opt.	30k	15.30	0.528	0.467	0.478	0.451	1966	3762
MVSplat	Feed-forward	0	16.44	0.741	0.271	0.293	0.717	3106	6018
DepthSplat	Feed-forward	0	16.13	0.760	0.260	0.291	0.707	3471	6730
MeshSplat	Feed-forward	0	16.55	0.732	0.270	0.275	0.745	2740	5339
Mesh-based methods									
MeshSplatting	Per-scene opt.	30k	15.92	0.449	0.585	0.462	0.479	**22**	**23**
LoomMesh (ours)	Feed-forward	0	**22.22**	**0.802**	**0.187**	**0.147**	**0.895**	131	265

GH200 GPUs, taking approximately 30 hours. Further training and evaluation details are provided in Appendix A.

Datasets. We train and evaluate on RealEstate10K (Zhou et al., 2018) with the pixelSplat (Charatan et al., 2024) split of 67,477 training and 7,289 test scenes, and evaluate on the 20-scene test subset of MeshSplat (Chang et al., 2026). Images are resized to 256×256, with two context views. For cross-dataset generalisation, we evaluate the same model on a random subset of DL3DV (Ling et al., 2024) without fine-tuning.

Metrics. We report PSNR, SSIM, and LPIPS (Zhang et al., 2018) on mesh renderings. For geometric quality, we measure per-component L1 error and mean cosine similarity between normals from mesh-rendered depth and pseudo-reference normals estimated by Metric3D-v2 (Hu et al., 2024) from the target images. We also report vertex and face counts to quantify mesh complexity.

Competitors. We compare LoomMesh with two categories of reconstruction methods: (1) *Gaussian-based methods*, including per-scene optimisation methods 3DGS (Kerbl et al., 2023) and 2DGS (Huang et al., 2024), and feed-forward methods MVSplat (Chen et al., 2024), DepthSplat (Xu et al., 2025), and MeshSplat (Chang et al., 2026). For the feed-forward baselines, we use their released RealEstate10K checkpoints for two-view reconstruction at 256×256 resolution. (2) *Mesh-based methods*, represented by MeshSplatting (Held et al., 2026a), which directly optimises a triangle mesh per scene. All methods use the same two context views and are evaluated as meshes. Gaussian outputs are converted through TSDF fusion (Curless & Levoy, 1996) of rendered depth maps. Further details are provided in Appendix A.

4.2 Main Results

Mesh Reconstruction We present the quantitative results in Table 1, evaluating both the geometric quality of the exported meshes and their novel-view synthesis quality using mesh-based rendering. LoomMesh achieves the best novel-view synthesis quality across all three rendering metrics. Compared with Gaussian-based feed-forward methods, it improves PSNR by 5.67 dB over MeshSplat, while directly producing a mesh without post-hoc surface extraction. LoomMesh achieves a 6.30 dB PSNR gain without the 30k iterations of per-scene optimisation. These results demonstrate the benefits of direct feed-forward mesh prediction under sparse two-view observations. In terms of geometric quality, LoomMesh also achieves the lowest normal L1 error and highest cosine similarity. Compared with MeshSplat, the strongest baseline on both metrics, it reduces L1 error from 0.275 to 0.147 and increases cosine similarity from 0.745 to 0.895, indicating better agreement with the pseudo-reference surface normals. These improvements are achieved with substantially fewer vertices and faces than the meshes extracted from Gaussian-based baselines.

Qualitative comparisons in Figure 3 show both textured meshes and shaded geometry. LoomMesh preserves clearer appearance details and more coherent scene structure, particularly around the fireplace in the first scene and the floor and walls in the second. The feed-forward Gaussian baselines recover the overall scene layout, but their extracted meshes exhibit distorted boundaries and irregular surfaces. Under the same two-view setting, per-scene methods show more pronounced missing

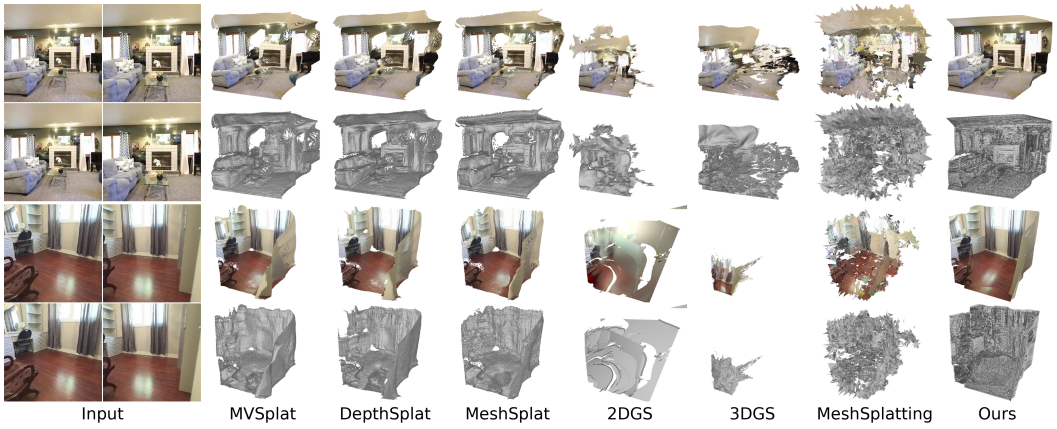

Figure 3: **Reconstructed geometry on RealEstate10K** test scenes, shown textured and as shaded geometry.

Table 2: **Zero-shot generalization to DL3DV** (256×448, two context views) with models trained on RealEstate10K only. All methods are evaluated as meshes.

	Novel-view synthesis			Normal accuracy		Mesh size (K)	
Method	PSNR↑	SSIM↑	LPIPS↓	L1↓	Cos.↑	Vertices↓	Faces↓
MVSplat (TSDF)	9.94	0.405	0.475	0.324	0.686	4677	9074
DepthSplat (TSDF)	9.47	0.426	0.473	0.305	0.706	5466	10520
MeshSplat (TSDF)	9.85	0.395	0.488	0.332	0.680	4690	9090
LoomMesh (ours)	**18.92**	**0.715**	**0.221**	**0.209**	**0.839**	**229**	**655**

geometry or fragmentation. In comparison, our method reconstructs more continuous surfaces while retaining the appearance of the input scenes.

Cross Dataset Generalization To thoroughly evaluate our method, we conduct cross-dataset generalisation experiments on DL3DV using models trained on RealEstate10K, without fine-tuning. As shown in Table 2, LoomMesh outperforms the Gaussian-based feed-forward baselines across all rendering and normal accuracy metrics while using substantially fewer vertices and faces. These results demonstrate that LoomMesh generalises beyond the training dataset, given that DL3DV's scenes are larger and more varied than RE10K, retaining its advantages in both mesh rendering and geometric quality.

4.3 Ablation Study

We conduct ablation studies on RealEstate10K to examine the topology reasoning operator (TRO) and its inputs. Table 3 reports novel-view synthesis and geometric quality.

Components of the topology reasoning operator. Our TRO is designed to refine vertex geometry and connectivity jointly. To examine this design, we first remove the entire operator (a). PSNR drops from 22.09 to 19.29 dB, while normal L1 error rises from 0.145 to 0.172. We then disable depth and connectivity refinement separately. Removing depth refinement (b) yields 19.33 dB PSNR, close to the variant without the TRO: predicting connectivity alone cannot compensate for inaccurate vertex positions. Removing connectivity refinement (c) also degrades rendering and geometry, with PSNR falling to 21.17 dB and normal L1 error rising to 0.169. Although depth refinement recovers much of the rendering quality, both readouts are needed to achieve the full model's geometric accuracy. Finally, without conductance gating (d), PSNR drops to 21.40 dB and normal L1 error increases to 0.156, showing the value of weighting neighbour messages according to their predicted reliability.

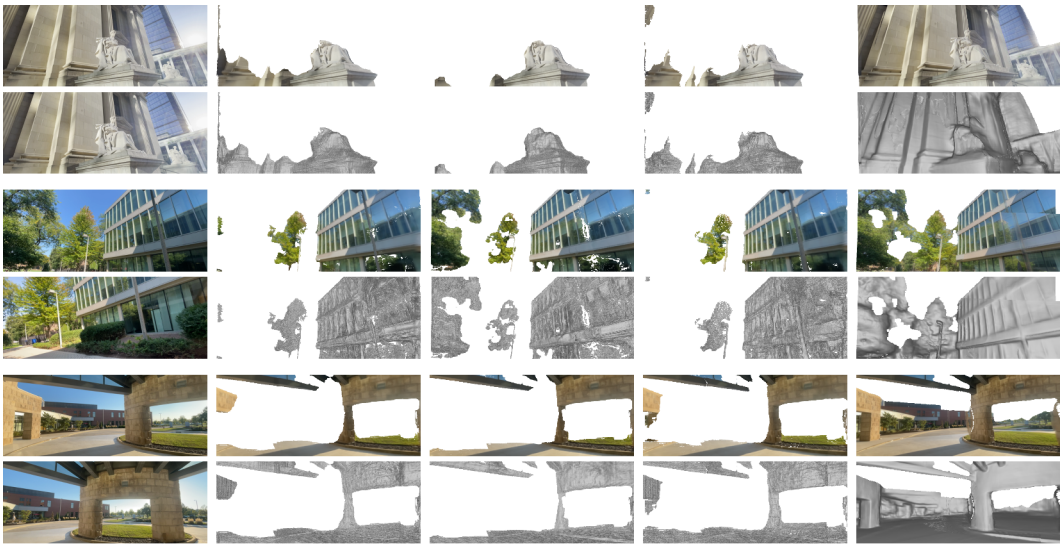

Figure 4: **Zero-shot reconstructions on DL3DV scenes** (256×448, two context views) using models trained on RealEstate10K only, shown textured and as shaded geometry.

Table 3: **Ablation of the topology reasoning operator (TRO)** on RealEstate10K (256×256, two input views). Best results are in **bold**.

	Novel-view synthesis			Geometric quality	
Variant	PSNR↑	SSIM↑	LPIPS↓	L1↓	Cos.↑
(a) w/o TRO	19.29	0.663	0.270	0.172	0.876
(b) w/o depth refinement	19.33	0.665	0.269	0.170	0.879
(c) w/o connectivity refinement	21.17	0.739	0.225	0.169	0.884
(d) w/o conductance gating	21.40	0.756	0.216	0.156	0.891
(e) w/o DINO descriptors	21.09	0.755	0.222	0.153	0.895
(f) w/o depth descriptors	20.78	0.743	0.227	0.153	0.895
(g) **Full model**	**22.09**	**0.788**	**0.196**	**0.145**	**0.898**

Node descriptors. We next examine the information used to initialise vertex states and included in message passing by removing by removing DINO descriptors (e) and depth descriptors (f) separately. PSNR drops to 21.09 and 20.78 dB, respectively. Both variants increase normal L1 error to 0.153. The results indicate that neither appearance features nor depth information can be discarded without weakening the topology reasoning operator.

5 Conclusion

We present LoomMesh, a feed-forward method that reconstructs an explicit triangle mesh from two posed images. Using predicted depths, it unprojects image pixels into 3D vertices and learns which candidate connections form the surface. A shared topology reasoning operator jointly refines vertex geometry and connectivity by reading depth-matching evidence and gating messages between neighbours. The method requires neither per-scene optimisation nor a separate surface-extraction stage. It is trained without ground-truth meshes using photometric and geometric losses. Experiments show higher mesh-based novel-view rendering quality and better surface-normal consistency than the evaluated per-scene mesh splatting and feed-forward mesh-conversion baselines.

AI USE STATEMENT

We used large language models (LLM) for text polishing and grammar refinement.

REPRODUCIBILITY STATEMENT

Implementation and training details are provided in the appendix. The main paper describes the model architecture, training objective, datasets, baselines, and evaluation metrics used in the experiments. The appendix further reports the remaining details of: optimisation settings, the loss weights, evaluation-set construction, and ablation protocols. Additionally method details are also given more in details which includes the graph construction, the update cell and the evidence lookup of the operator, and the renderer's traversal and gradients.

REFERENCES

Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 35–45, 2019.

Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

Xavier Bresson and Thomas Laurent. Residual gated graph ConvNets. *arXiv preprint arXiv:1711.07553*, 2017.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2716–2724, 2026.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14124–14133, 2021.

Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

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 (ECCV)*, pp. 370–386. Springer, 2024.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 303–312. ACM, 1996.

Boris Delaunay. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles*, (6):793–800, 1934.

Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. MILo: Mesh-in-the-loop Gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics*, 44(6), 2025.

Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. MeshSplatting: Differentiable rendering with opaque meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7320–7329, 2026a.

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Daniel Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In *International Conference on 3D Vision (3DV)*, pp. 1248–1257, 2026b.

Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46, 2024.

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

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42:139:1–139:14, 2023.

Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8, 2007.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations (ICLR)*, 2026.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22160–22169, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pp. 405–421. Springer, 2020.

Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.

Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12179–12188, 2021.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

Kaifeng Sheng, Zheng Zhou, Yingliang Peng, and Qianwei Wang. 2D triangle splatting for direct differentiable mesh training. *arXiv preprint arXiv:2506.18575*, 2025.

Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C. Lin, and Yi Zhou. DMesh: A differentiable mesh representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 12035–12077, 2024.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pp. 402–419. Springer, 2020.

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 311–318. ACM, 1994.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699, 2021.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16453–16463, 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. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3D Gaussian splatting. In *European Conference on Computer Vision (ECCV)*, pp. 1–19. Springer, 2024.

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

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):65:1–65:12, 2018.

A TRAINING AND EVALUATION DETAILS

Optimisation. We use AdamW (Loshchilov & Hutter, 2019) with weight decay 0.01 and three learning rates: 2×10^{-4} for all parameters trained from scratch, 2×10^{-6} for the pre-trained ViT-L, and 1×10^{-4} for the two vote heads. The schedule is one-cycle with a 1% warm-up and cosine annealing, and the gradient norm is clipped at 0.5. We train for 100k iterations with 16 scenes per batch (4 per GPU on four GH200 GPUs), each with two context views 45–135 frames apart and four target views; training takes 30 hours. The output layer of the depth head is zero-initialised, so training starts from a uniform matching posterior.

Losses. The weights in equation 13 are $\lambda_{\mathrm{lp}} = 0.05$, $\lambda_{\mathrm{n}} = 0.2$, $\lambda_{\mathrm{gm}} = 0.05$ and $\lambda_{\mathrm{w}} = 0.05$, constant throughout training. After 5k iterations, the photometric L_2 loss becomes a Huber loss, so pixels the mesh cannot yet explain do not dominate the gradient.

The rendered normal $\mathcal{L}_{\mathrm{normal}}$ is composited from the face normals of each pixel's nearest hits with the same weights as colour, against the target normals computed from the depth maps using central-differences where the depth maps are obtained from DepthAnythingV3 (Lin et al., 2026). $\mathcal{L}_{\mathrm{gm}}$ follows the gradient-matching loss of MiDaS (Ranftl et al., 2022). It compares how the rendered and teacher depths change across the image, through their log-ratio $R = \log(D/A) - \log z_{\mathrm{T}}$, with z_{T} the teacher depth:

$$\mathcal{L}_{\mathrm{gm}} = \frac{1}{S} \sum_{s=1}^{S} \Big(\overline{|\partial_x R^s|} + \overline{|\partial_y R^s|} \Big), \tag{14}$$

where R^s is R average-pooled by 2^{s-1} over valid pixels (a coarse pixel is valid if more than half of its four children are), ∂_x and ∂_y are differences between horizontally and vertically adjacent valid pixels, the bars are means over valid pairs, and $S = 4$. Where both depths are smooth, or both step by the same ratio, R is locally constant and the loss is zero. It fires only where one depth steps and the other does not: across an edge the mesh has missed, such as a face stretched over an occlusion boundary, or across one it has invented. A global scale between the two depths adds a constant to R, which cancels in the differences, so the loss needs no scale alignment and leaves absolute depth to the cost volume and the photometric loss. $\mathcal{L}_{\mathrm{warp}}$ saturates at a log-depth gap of 0.3, so true occlusions receive no gradient. $\mathcal{L}_{\mathrm{warp}}$ saturates at a log-depth gap of 0.3, so true occlusions receive no gradient.

Evaluation. On RealEstate10K we use the 20-scene mesh-evaluation split of MeshSplat, with two context and three target views per scene at 256×256. On DL3DV we use 20 DL3DV-Benchmark scenes with two context views 8–33 frames apart and three target views between them, resized and centre-cropped to 256×448. For MVSplat, DepthSplat and MeshSplat, the Gaussians predicted from the two context views are rendered at the two context and three target cameras, following MeshSplat, and the rendered depth and colour are fused into a TSDF (Curless & Levoy, 1996) with voxel size 0.005, truncation 0.1 and a maximum depth of 5.0 on RealEstate10K, scaled per scene on DL3DV. Marching cubes extracts the mesh, and we keep its 50 largest connected components. 3DGS and 2DGS are optimised per scene on the two context views for 30k iterations and meshed in the same way; MeshSplatting is optimised per scene on the two context views and exports its own mesh.

Surface metrics. Metric3D-v2 (ViT-L) predicts the reference normals on the real target images, at 256×256 for RealEstate10K and 256×448 for DL3DV. Each mesh is ray-cast from the same cameras, and its normals are finite differences of the ray-cast depth, computed identically for every method. Where both normals are valid, the cosine is the mean dot product of the unit normals and the L1 is the mean absolute difference per normal component; scores are averaged over pixels, then views, then scenes.

B METHOD DETAILS

B.1 WHY TRAINING STALLS WITHOUT THE MODULE

Without the topology update module, depth is the upsampled soft-argmax $\hat{d}^0$, and every face opacity is $\alpha_f = \min_{e \in \mathcal{E}(f)} \sigma(\eta_e)$: a fixed function of geometry that passes no gradient (App. B.3). A Gaussian's depth is refined independently of its neighbours, but a mesh vertex is tied to its neighbours by the faces around it, which are kept or cut by its depth relative to theirs. Faces then stretch across occlusion boundaries, and three effects keep them there.

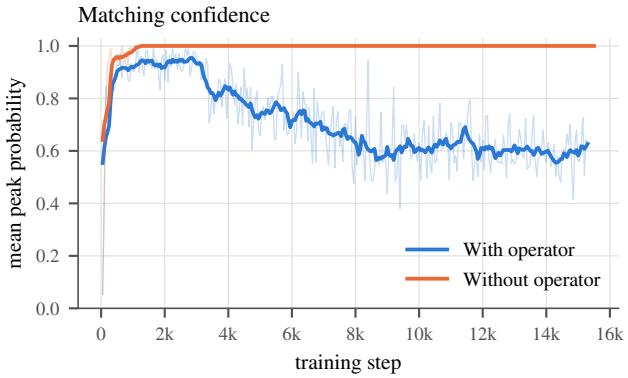

Figure 5: **Matching confidence during training**, the mean peak probability of the matching posterior. Without the operator, the posterior collapses onto a single candidate within about 1k steps and stays there, so the soft-argmax passes no further gradient to the matcher. With the operator, it also sharpens early, but from about 3k steps the matcher keeps changing, and its mean peak probability settles near 0.6.

Faces stretch across boundaries. The depth head starts from zero logits (App. A), so at step 0 the posterior is uniform, the depth is the same constant at every pixel, and every grid face is kept. Depth then develops gradually and at $1/s$ of the input resolution, so an occlusion boundary first appears as a smooth ramp. The gate reads a ramp as a slanted surface, whose residual is near zero, so the faces across it are kept and stretch. The photometric loss cannot cut such a face, since α_f passes no gradient. Nor does it pull the face's vertices apart: their depths receive gradient mainly through the faces as rendered, and the cut that a wider gap would trigger is invisible to that gradient. The vertices are therefore fitted to the stretched face rather than to their surfaces.

What the module changes. The face vote is a differentiable function of the module's state, so the photometric loss can lower the opacity of a stretched face directly, without a path through geometry. The gated messages let a vertex on a stretched face read the depth of its own surface, and the depth step moves it there. The lookup keeps passing gradient to the matching scores after saturation (App. B.5), and this gradient arrives mainly through the depth step (Sec. 4.3). The whole operator has 0.11M parameters, 0.03% of the model.

B.2 CANDIDATE GRAPH

Section 3.2 restricts the meshes the network can output to the triangles of a candidate graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that is rebuilt at every forward pass. Here we give its exact construction, the properties the rest of the method relies on, and its size in practice.

Grid links. Within each view, every 2×2 block of pixels is split into two triangles along its diagonal from (i, j) to $(i{+}1, j{+}1)$, as in depth-map meshing. Each pixel is therefore linked to its right, lower and lower-right neighbours, and has at most six grid links. Grid links alone close exactly these two triangles in every block and no others, so every other candidate face uses at least one cross-view link.

Cross-view links. Each vertex $\mathbf{v}_{a,p}$ of view a is projected into every other view b with the pinhole camera of b, and linked to the pixel $q = \lfloor \Pi_b(\mathbf{v}_{a,p}) \rfloor$ it lands in and to the four neighbours of q, provided the point lies in front of camera b and inside its image. Links are generated in both directions and merged into undirected edges. Since projection is many-to-one, many pixels of view a can land in one pixel of view b, so the number of cross-view links at a vertex is unbounded. We therefore rank the cross-view links at each vertex by their 3D length and keep the 8 shortest, dropping a link that exceeds this cap at either endpoint. Every vertex thus has degree at most $6 + 8 = 14$.

Hard filters outside the gate. An edge longer than 32 resolvable units, measured against the larger spacing ϕ of its two endpoints (App. B.3), is discarded before faces are enumerated. Since the spacing is at least the pixel footprint, this bound never removes a grid link on a surface slanted by less than about 88°; it removes only links spanning more than about 16 bins, which the gate would cut unless a vote held them.

B.3 Topology gate

Gate equation. Each candidate edge carries the conductance $k_e = \sigma(\eta_e - g_e)$ of equation 6, which compares the geometric evidence η_e of equation 5 with a learned vote g_e; η_e is positive when the residual is within the tolerance and negative outside it. Two scalars in η_e are fixed throughout and shared by all edge types: the tolerance $\rho_\star = 4$ and the temperature $\tau = \frac{1}{4}$. The normal $\mathbf{n}_i$ is the cross product of two tangents along the image axes. Each tangent is a one-sided difference to a lattice neighbour, taking the shorter of the two sides, so a vertex on the rim of a fold takes the tangent of its own surface. In all conductances the sigmoid is sharpened by a fixed temperature $\tau = \frac{1}{4}$, $k_e = \sigma\big((\eta_e - g_e)/\tau\big)$, which the main text omits for brevity. With no vote, $k_e = 1/\big(1+(\rho_e/\rho_\star)^{1/\tau}\big)$, so $\rho_\star$ sets where the gate switches ($k_e = 0.5$) and $1/\tau$ how sharply.

Learned votes. Both votes, g_e in equation 8 and g_e^{face} in equation 10, are two-layer MLPs with zero-initialised last layers, and their outputs are bounded as $\hat{g}\tanh(\cdot/\hat{g})$ with $\hat{g} = 16$. A vote has the same effect as moving the tolerance from $\rho_\star$ to $\rho_\star e^{-\tau g_e}$, so the effective tolerance stays within $[\rho_\star e^{-4}, \rho_\star e^{4}] \approx [0.07, 218]$ resolvable units.

B.4 Cell update and the exchange

Inputs. ℓ_i enters mean-centred, with the mean taken over all vertices in the batch, and c_i is the peak of the vertex's posterior; both are taken without gradient. $\mathbf{f}_i$ is the final-layer feature of the monocular ViT-L, which Depth Anything V2 initialises from DINOv2 and which trains with the backbone at the low learning rate of App. A; the module reads it without gradient, resampled bilinearly to the coarse grid and ℓ_2-normalised, and learns only the projection $\mathbf{P} \in \mathbb{R}^{32\times 1024}$. The edge family in $\mathbf{e}_{ij}$ is one of horizontal, vertical, diagonal and cross-view, and the residual and length are those of the initial depth. All MLPs in the module have two layers, hidden width 64 and GELU activations.

Update cell. The update cell of equation 7 is a standard GRU cell (Cho et al., 2014), as implemented in PyTorch, applied to every coarse vertex. It receives the vertex's three sources as one vector $\mathbf{a}_i = [\mathbf{E}_i, \mathbf{M}_i, \mathbf{x}_i] \in \mathbb{R}^{103}$: 5 evidence scores (equation 9), 64 message channels (equation 8) and the 34 own inputs $\mathbf{x}_i = [\ell_i, c_i, \mathbf{P}\mathbf{f}_i]$. Its state starts from $\mathbf{h}_i = \mathbf{W}_{\text{in}}\mathbf{x}_i \in \mathbb{R}^{64}$, an embedding of the vertex's own inputs. Omitting biases,

$$\begin{aligned} \mathbf{r}_i &= \sigma\big(\mathbf{W}_r\mathbf{a}_i + \mathbf{U}_r\mathbf{h}_i\big), \quad \tilde{\mathbf{h}}_i = \tanh\big(\mathbf{W}_h\mathbf{a}_i + \mathbf{r}_i \odot \mathbf{U}_h\mathbf{h}_i\big), \\ \mathbf{u}_i &= \sigma\big(\mathbf{W}_u\mathbf{a}_i + \mathbf{U}_u\mathbf{h}_i\big), \quad \mathbf{h}'_i = \mathbf{u}_i \odot \mathbf{h}_i + (1 - \mathbf{u}_i) \odot \tilde{\mathbf{h}}_i, \end{aligned} \tag{15}$$

where σ is the logistic sigmoid and $\odot$ the element-wise product. The candidate $\tilde{\mathbf{h}}_i$ is a new state proposed from the sources, and the reset gate $\mathbf{r}_i$ decides how much of the current state it may use. The update gate $\mathbf{u}_i$ then decides, for each of the 64 channels, how much of the current state survives: $\mathbf{u}_i \to 1$ keeps $\mathbf{h}_i$, and $\mathbf{u}_i \to 0$ takes the candidate. Using a GRU cell as the node update of a graph network follows gated graph neural networks (Li et al., 2016).

Depth step and face vote. The head φ_z in equation 10 is a two-layer MLP ($64 \to 64 \to 1$) with a zero-initialised last layer, so the first step is exactly zero. The bound $\delta_{\max} = 0.05$ in log inverse

depth corresponds to about 5% of the depth at any scale, and after the step ℓ_i is clamped to the hypothesis range $[\log d_{\min}, \log d_{\max}]$. For the face vote, the states $\mathbf{h}^{(1)}$ are upsampled bilinearly to full resolution, and φ_{face} is evaluated on every full-resolution candidate edge with that edge's family and residual at the refined depth; its bound and initialisation are those of App. B.3.

B.5 Evidence lookup

The lookup follows the correlation lookup of RAFT (Teed & Deng, 2020): the matching posterior plays the role of RAFT's correlation volume, the log inverse depth ℓ_i that of its flow, and the depth step of equation 10 that of its residual flow update. The lookup of equation 9 reads the matching posterior of vertex i around its current estimate. We write $p(n \mid x)$ for the posterior over the $D = 128$ candidates $n = 0, \dots, D-1$, and s_n for the matching scores it is the softmax of. The candidates are uniform in inverse depth, $d_n = d_{\min} + n\,\Delta\sigma$, where $\Delta\sigma = (d_{\max} - d_{\min})/(D-1)$ is the bin width of equation 5. The estimate sits at the fractional index b_i, and at a fractional index t the log-posterior is interpolated linearly between the two neighbouring candidates:

$$\log p(t \mid x) = \big(\lfloor t \rfloor + 1 - t\big) \log p\big(\lfloor t \rfloor \mid x\big) + \big(t - \lfloor t \rfloor\big) \log p\big(\lfloor t \rfloor + 1 \mid x\big). \tag{16}$$

Read positions outside the range are clamped to $[0, D-1]$. The estimate itself always lies inside it, since it starts as a soft-argmax over the candidates and the depth step is clamped (equation 10). With $w = 2$, $\mathbf{E}_i$ holds five values one bin apart, centred on the estimate.

Gradients. The read passes gradient back along two paths. Through the position, it passes the slope of the log-posterior, $\log p(\lfloor t \rfloor{+}1 \mid x) - \log p(\lfloor t \rfloor \mid x)$, which reaches ℓ_i through $\partial b_i / \partial \ell_i = e^{\ell_i}/\Delta\sigma$. Through the values, it reaches the matching scores directly:

$$\frac{\partial \log p(n \mid x)}{\partial s_m} = \mathbf{1}[m = n] - p(m \mid x), \qquad \text{whereas} \qquad \frac{\partial \hat{d}}{\partial s_m} = p(m \mid x)\,\big(d_m - \hat{d}\big) \tag{17}$$

for the soft-argmax $\hat{d} = \sum_n p(n \mid x)\, d_n$.

B.6 Renderer details

Traversal. Faces are binned to 16×16-pixel tiles by their projected bounding boxes and sorted once by a conservative per-face depth key. This sort only fixes the traversal order: each pixel keeps its $k = 8$ nearest hits in a buffer ordered by exact t.

Gradient path. The backward pass treats which faces enter a pixel's buffer, and their order, as locally constant, as a z-buffer treats its argmin, and differentiates everything else exactly. Each output $X \in \{C, D, A\}$ of equation 12 gives hit i the front-to-back compositing gradient

$$\frac{\partial X}{\partial a_i} = T_i\, x_i - \frac{1}{1 - a_i}\Big(\sum_{j>i} T_j a_j\, x_j + T_{k+1}\, x_{\text{bg}}\Big), \tag{18}$$

with $x_i = \mathbf{c}_i$, t_i and 1 for C, D and A, and $x_{\text{bg}} = \mathbf{c}_{\text{bg}}$ for colour and 0 otherwise; the cap at $1 - \epsilon$ passes gradients through, as in 3DGS. The product $a_i = \alpha_{f_i} I_{f_i}$ splits this gradient in two. The share $I_{f_i}\, \partial\mathcal{L}/\partial a_i$ reaches the face opacity and, through it, the face vote (App. B.3). The share $\alpha_{f_i}\, \partial\mathcal{L}/\partial a_i$ reaches the window, whose derivative $\partial I_f / \partial \lambda_m = (h_m/\sigma_w) \prod_{j \neq m} r_j$, with r_m the m-th factor of equation 11, is non-zero only on the band, where $0 < r_m < 1$; σ_w and the altitudes h_m are held constant.

C More Visual Comparisons

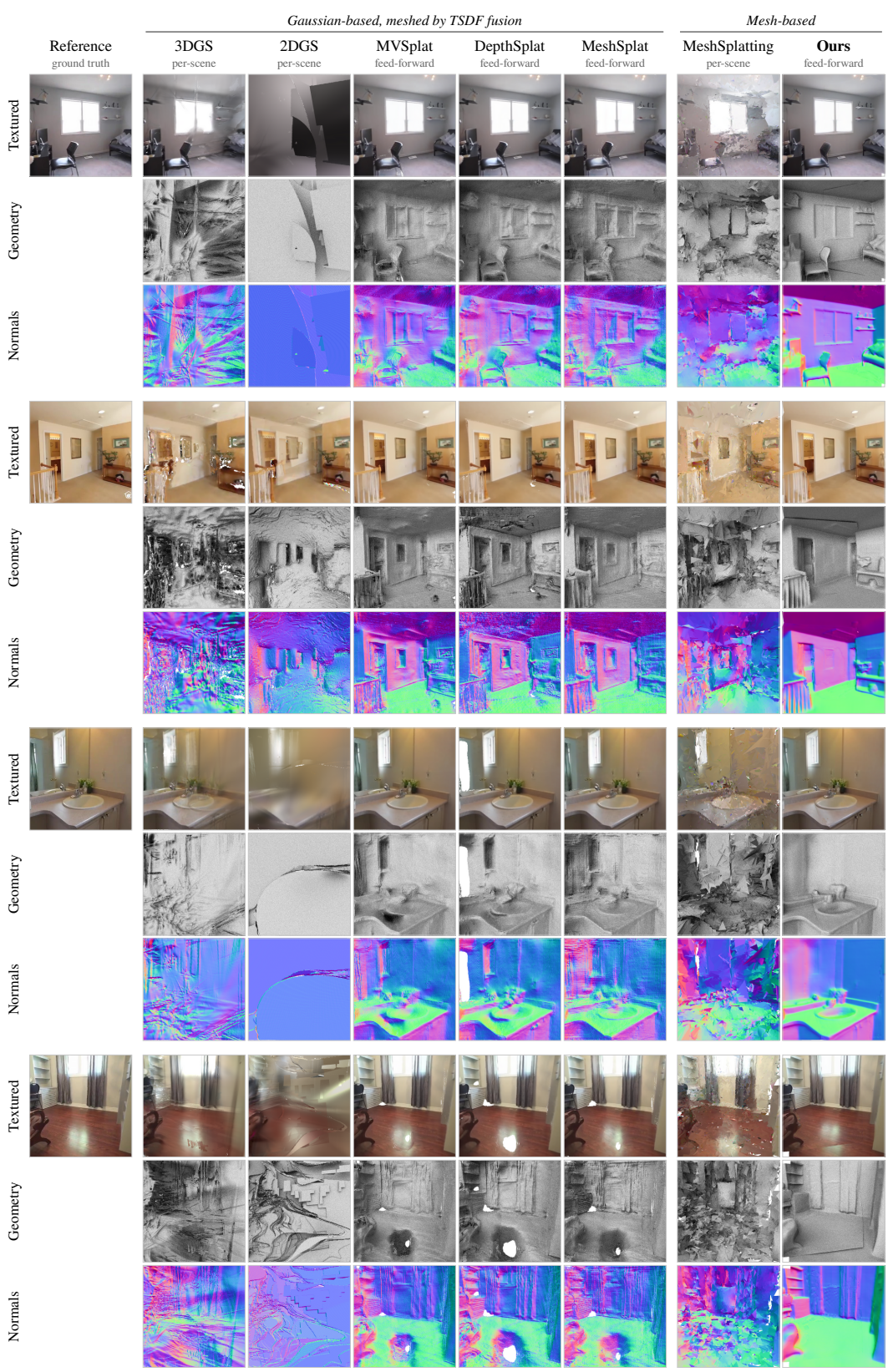

Figure 6: **Mesh comparison on RealEstate10K** test scenes at a target view, shown textured, as shaded geometry and as normals. Gaussian baselines are meshed by TSDF fusion.

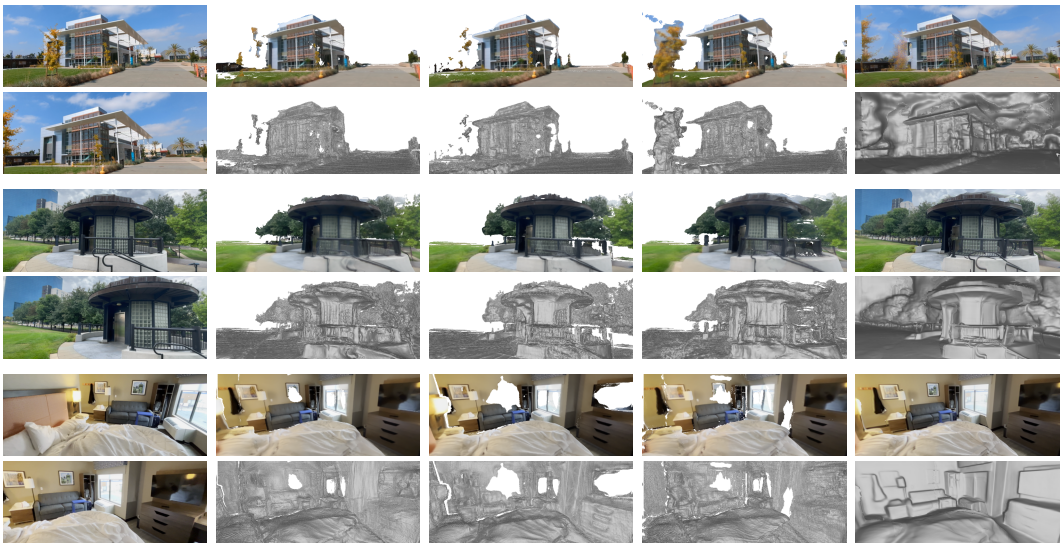

Figure 7: **Additional mesh comparisons on DL3DV (zero-shot).** All models use their RealEstate10K checkpoints and the two 256×448 input views on the left. Each mesh is rendered at a held-out target view, textured (top row of each scene) and as shaded geometry (bottom row); white means no surface. Baseline meshes are fused from their rendered depth with MeshSplat's TSDF protocol; ours are predicted directly.

D Limitations and Future Work

LoomMesh places one vertex at every pixel of every input view, so the mesh follows the image grid rather than the geometry. Its size grows with the input resolution. Secondly, our current model is still deterministic rather than generative, which limits its ability to handle occluded regions. Future work includes adaptive vertex density: predicting where vertices are needed, so that the size of the mesh follows the complexity of the scene rather than the size of the images. Also combining the reconstruction with a generative prior, such as a multi-view or video diffusion model, could better reason about occluded or unobserved regions.

E Video Demo

We provide a video demo of a LoomMesh mesh used directly in a physics simulation. We import the mesh predicted for a RealEstate10K test scene into Blender as exported and use it as the static collider for rigid spheres, boxes and cylinders, simulated with Blender's Bullet physics engine. The objects fall onto the reconstructed floor and furniture, collide with them and come to rest. RealEstate10K poses have no gravity direction or metric scale, so we align the up axis with a fitted floor plane and set the scale from an assumed camera height of 1.6 m. The mesh is not otherwise processed: no hole filling, remeshing or smoothing. Please refer to the supplementary material for the video.

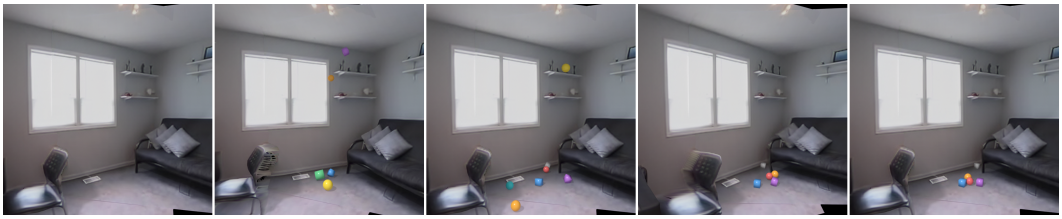

Figure 8: Frames from the supplementary video, left to right ($t = 0$ to 8 s): rigid bodies dropped onto a LoomMesh mesh and simulated in Blender, with the predicted mesh as the collider.