T3LESCOPE 🔭: ARBITRARY-RESOLUTION HIGH-FIDELITY GENERATIVE SURFACE RECONSTRUCTION FROM IMAGES

Anonymous authors
Paper under double-blind review

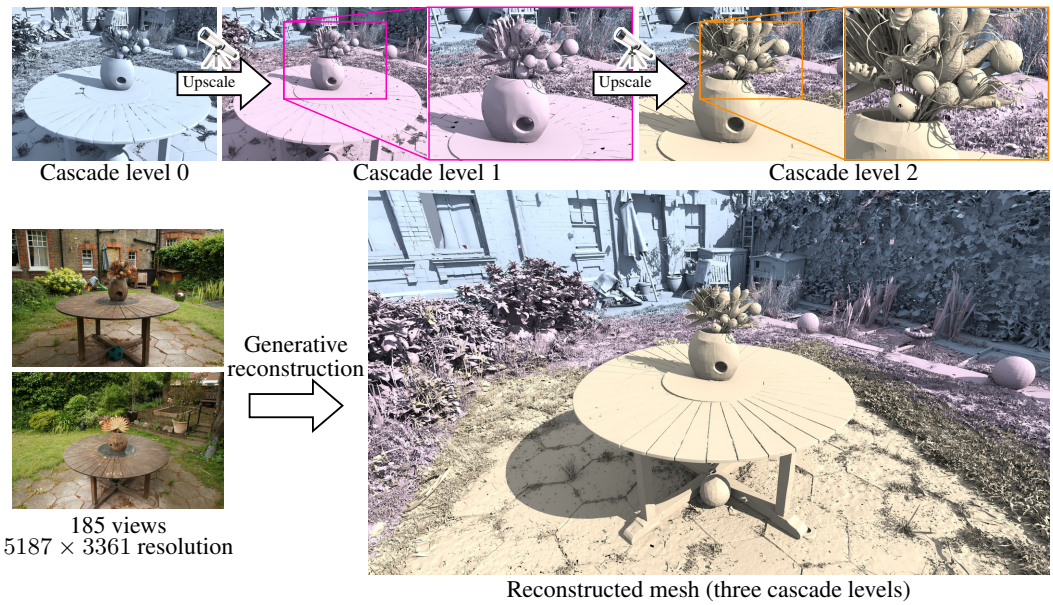


Figure 1: **High-fidelity mesh reconstruction without per-scene optimization.** Our method takes multi-view images as input and directly outputs a high-resolution mesh with mesh flow models. We adopt a coarse-to-fine cascade: the entire scene is generated at a coarse resolution first, and regions of interest are then upscaled. The cascade layout and the mesh resolution are chosen at inference time. Colors mark the cascade level each surface was generated at.

ABSTRACT

We reconstruct high-fidelity 3D scene meshes from posed multi-view images without per-scene optimization, across scales ranging from single objects to large outdoor scenes. Per-scene optimization methods lack the learned 3D prior needed when observations are sparse or surfaces are glossy or transparent. Existing generative methods leverage such priors to complete geometry in sparsely observed regions, but typically operate at a fixed resolution over a limited spatial extent, trading spatial coverage against detail. Reconstructing a large scene therefore often requires partitioning it into independently processed overlapping local regions, making it difficult to maintain global geometric consistency. To address these issues, we propose T3lescope, which applies a single fixed-resolution generator across scene scales in an inference-time coarse-to-fine cascade. A coarse level establishes the scene layout, and finer levels perturb and denoise geometry inherited from the coarser level within progressively finer spatial cells to recover surface detail. The model is trained on individual cells at multiple scales and shares its weights across all levels, so no hierarchy is fixed during training, and the number of levels, cell scales, and cell locations are determined at inference time. On indoor, outdoor, and city-scale scenes, T3lescope outperforms feed-forward and

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.

generative baselines, matches or surpasses per-scene optimization, and recovers fine structures as well as glossy and transparent surfaces. These results show that our method generalizes across diverse scenes, view counts, and image resolutions.

1 INTRODUCTION

Reconstructing high-fidelity 3D meshes from posed multi-view images is important for a broad range of real-world applications, including robotic perception, simulation, digital twins, and immersive environments. These applications require both fine surface detail and globally consistent scene geometry, across targets ranging from single objects to large outdoor scenes. Existing approaches broadly fall into three categories: per-scene optimization, feed-forward prediction, and generative reconstruction. Optimization-based methods fit a scene-specific 3D representation to the input images and can recover accurate surfaces from dense observations (Wang et al., 2021; Huang et al., 2024; Guédon et al., 2025). However, they require optimization for every scene and, lacking a learned 3D prior, fail under sparse views, occlusion, or glossy, reflective, and transparent surfaces.

Feed-forward methods address the cost of per-scene optimization by learning priors across many scenes and directly predicting geometry such as depth or point maps (Wang et al., 2024; Lin et al., 2026; Keetha et al., 2026). However, their outputs are typically image-aligned and must be fused across views to form a complete mesh, where cross-view inconsistencies can introduce noise, holes, or duplicate surfaces. Generative reconstruction addresses both issues by conditioning a learned 3D generator on the input views. Like feed-forward methods, it avoids per-scene optimization, while its learned prior directly predicts a complete 3D representation, including weakly constrained surfaces (Xiang et al., 2025; 2026). Its main limitation is scale: current 3D generators typically operate on a fixed voxel grid at a single scene scale, trading spatial coverage against geometric detail. GenRecon (Schmid et al., 2026), the closest prior work, extends such a fixed-resolution generator to large scenes using overlapping local cells with MultiDiffusion (Bar-Tal et al., 2023). While this increases spatial coverage, distant regions interact only indirectly through overlaps, making global geometry difficult to maintain. Coarse-to-fine 3D generators offer another solution, but existing approaches fix their hierarchy, spatial extent, and output resolution during training (Ren et al., 2024a;b). These limitations motivate a different question: can a single fixed-resolution generator be composed across scene scales at inference time, from global structure to progressively finer detail?

To this end, we propose T3lescope, an inference-time coarse-to-fine cascade that reuses the same generator across multiple scene scales. A coarse level reconstructs the full scene and establishes its global structure, while each finer level refines inherited geometry by halving the cell size, perturbing it with noise, and denoising it to recover finer surface detail. The same weights are shared across all levels and trained on individual cells at multiple scene scales, without paired coarse-to-fine supervision. Thus, cell size, layout, and cascade depth determine spatial coverage and output resolution at inference time. This flexibility, however, makes image conditioning nontrivial: cells vary in position and physical scale across scenes and cascade levels, while each level requires image features at a different resolution. To address this, we precompute reusable multi-resolution image features and condition each voxel using its projected location and projected size. This makes conditioning independent of absolute scene position and scale, allowing cells to be placed at arbitrary positions and scales while providing the detail required at each cascade level.

We evaluate T3lescope on two public benchmarks, ScanNet++ and Tanks and Temples, and on our own city-block dataset, under both sparse- and dense-view settings. It outperforms feed-forward and generative baselines and is competitive with per-scene optimization. It recovers fine structures as well as glossy and transparent surfaces. Results also show that our method generalizes across diverse scenes, view counts, and image resolutions (Figures 1 and 3). Ablations validate both parent initialization and image conditioning. Our contributions are:

- We propose an inference-time coarse-to-fine cascade that reuses one fixed-resolution 3D generator across scene scales, with coverage and resolution chosen at inference time.
- We develop a cell-independent multi-scale conditioning scheme that reuses image features across cells, levels, and sampling steps while allowing flexible cell placement and scale.
- We show that T3lescope outperforms feed-forward and generative baselines and is competitive with per-scene optimization on indoor, outdoor, and city-scale scenes.

2 Related Work

Mesh reconstruction. Mesh reconstruction from posed images has traditionally relied on multi-view stereo, which matches pixels across images and fuses the recovered geometry (Schönberger et al., 2016). Neural implicit and Gaussian-based methods instead optimize a 3D representation for each scene (Wang et al., 2021; Yu et al., 2022; Huang et al., 2024; Chen et al., 2025; Guédon et al., 2025; Gomez et al., 2026), with recent methods scaling to city-scale scenes (Liu et al., 2025). These approaches recover accurate geometry from dense views and can be regularized with learned monocular or geometric cues. However, they require per-scene optimization and tend to degrade under sparse observations or with challenging glossy and transparent surfaces. Feed-forward methods instead learn scene priors that generalize to unseen scenes and predict image-aligned depth or point maps in a single forward pass (Wang et al., 2024; Lin et al., 2026; Keetha et al., 2026; Guo et al., 2025). While they avoid per-scene optimization, their predictions must be fused across views, making the resulting mesh sensitive to cross-view inconsistencies. In contrast, our method directly generates the mesh with a 3D generator, without either per-scene optimization or multi-view fusion.

Generative 3D reconstruction. Recent 3D generative models directly produce high-resolution meshes through sparse structures and structured latents (Xiang et al., 2026). Pixel-aligned image conditioning improves geometric fidelity via explicit 2D–3D correspondence (Li et al., 2026a). GenRecon extends an object-level 3D generator to scene-scale reconstruction by tiling the scene with local overlapping cells and jointly sampling them with MultiDiffusion (Schmid et al., 2026; Bar-Tal et al., 2023). This extends generation beyond the model's native spatial extent, but each denoiser operates on a local cell, and distant regions interact only indirectly through neighboring overlaps. Instead, our cascade propagates global structure from the coarse reconstruction to each finer cell.

Hierarchical generation. Coarse-to-fine 3D generators reach high resolution with hierarchies whose levels and scales are fixed during training, often with a separate model at each level (Ren et al., 2024a;b; Meng et al., 2025). Methods for larger scenes scale spatial extent by progressively extending the generated region (Wu et al., 2024; Lu et al., 2025; Joshi et al., 2026; Li et al., 2026b; Meng et al., 2026). In 2D, cascaded diffusion models similarly train a separate model for each resolution (Ho et al., 2022), while training-free methods reuse the same model to progressively refine upsampled samples (Meng et al., 2022; Du et al., 2024). We adapt this idea to 3D. At each finer level, we reduce the cell size while keeping the generator resolution fixed. This allows the same model to be reused across scales, with the hierarchy configured at inference time.

3 Method

We reconstruct a mesh $\mathcal{M}$ from posed multi-view images with a 3D generative model, without per-scene optimization. Since the generator has a fixed resolution, we reconstruct the scene coarse-to-fine. The coarsest level captures the scene, and finer levels refine it (Section 3.2). At each level, we partition the scene into overlapping cells (Section 3.1) and apply the generator with corresponding image features (Section 3.4). We train a single model across scales (Section 3.3) and configure the hierarchy and spatial scale at inference time. Figure 2 shows the pipeline.

3.1 Single-Level Scene Generation

We adopt TRELLIS.2 (Xiang et al., 2026), a sparse-voxel mesh generator, as our generative model. It first generates the sparse structure (SS) of surface voxels and then generates a mesh over that structure. We refer to these two steps as the SS and mesh stages. The input is a set of posed images $\mathcal{I} = \{(I_i, K_i, T_i)\}_{i=1}^{N}$, where I_i is an RGB image, K_i its intrinsics, and T_i its camera pose. Both stages use 3D VAEs to encode their representations and Diffusion Transformers (Peebles & Xie, 2023) for flow matching. The generator operates on a fixed R^3 sparse voxel grid and is conditioned on DINOv3 (Siméoni et al., 2025) image features together with camera intrinsics and poses.

For scene generation, we follow GenRecon (Schmid et al., 2026) and partition the scene into overlapping cells that share a scene-wide latent updated by MultiDiffusion (Bar-Tal et al., 2023) in both the SS and mesh stages. At each flow step, we denoise each cell independently, then blend the overlapping predictions to update the shared scene latent. The world-space edge length s of each cell, the corresponding voxel spacing $\Delta = s/R$, and the cell placement are all set at inference time. Because

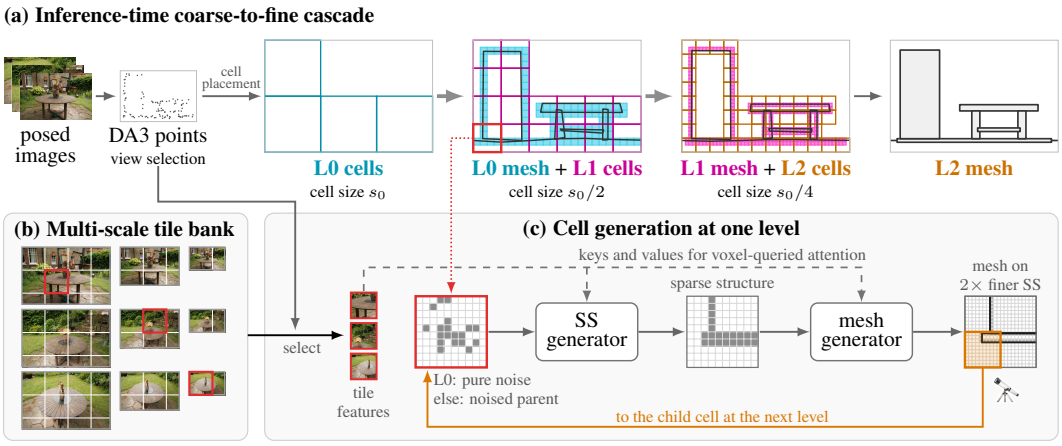


Figure 2: **Method overview.** Ln denotes level n. **(a)** DA3 points place the L0 cells and select their views. Each finer level halves the cell size and refines the mesh of the level above. **(b)** Each image is encoded once as a tile pyramid, and from each view a cell takes the tiles at the pyramid level that matches its voxel spacing. **(c)** All cells at all levels share the SS and mesh generators, which attend to these tiles. At finer levels, the SS stage starts from the noised parent (SDEdit).

the generator operates in canonical cell coordinates and receives neither absolute world position nor metric cell size, the same model can be applied to cells at arbitrary locations and scales.

However, GenRecon's MultiDiffusion-based inference extends generation beyond a single R^3 cell, while each denoiser still operates with only local cell context. Distant regions therefore interact only indirectly through neighboring overlaps, motivating our coarse-to-fine cascade, described in the next section, which propagates global context across scales.

3.2 Inference-Time Coarse-to-Fine Cascade

To overcome the limited context of single-level generation, we introduce a coarse-to-fine cascade. The coarsest level covers a large spatial extent and establishes the global structure, while each subsequent level refines the inherited geometry to recover finer surface detail. Each level halves the cell size, with cell size at level l given by $s_l = s_0/2^l$ for $l = 0, \dots, L-1$ (Figure 2(a)).

At every level, we derive the child sparse structure from the parent mesh decoder D_{shape}. The decoder upsamples the parent shape latent through subdivision stages, each doubling the spatial resolution. We use the output at resolution $2R$, where a child cell with half the parent edge length corresponds to an R^3 block of voxels, and take this block as the child's sparse structure. Because the parent geometry is estimated at a coarser resolution, subdivision alone leaves the surface coarse and misaligned with the finer child grid. This provides a noisy initialization, which we refine with the SS flow model at the child resolution (Figure 2(c)).

To refine this noisy initialization, we first encode the inherited sparse structure into the SS latent, $\bar{x}_0^{(l+1)} = A_{\text{SS}}\left(E_{\text{SS}}\left(\hat{O}_{\text{sub}}^{(l)}\right)\right)$, where $\hat{O}_{\text{sub}}^{(l)}$ is the R^3 block of the parent's subdivision occupancy that covers the child cell, E_{SS} is the SS encoder, and A_{SS} is the latent normalization used for flow matching. We then perturb $\bar{x}_0^{(l+1)}$ to an intermediate time t_s on the forward path, $x_{t_s} = (1 - t_s)\,\bar{x}_0^{(l+1)} + \sigma_{t_s}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$, with $\sigma_t = \sigma_{\min} + (1 - \sigma_{\min})t$, and integrate the SS flow back to $t = 0$ with K Euler steps over $t_s = \tau_K > \cdots > \tau_0 = 0$,

$$x_{\tau_{k-1}} = x_{\tau_k} - (\tau_k - \tau_{k-1})\, v_\theta(x_{\tau_k}, \tau_k; \mathcal{I}), \quad k = K, \dots, 1, \qquad \hat{x}_0^{(l+1)} = x_{\tau_0}. \tag{1}$$

This applies SDEdit (Meng et al., 2022) between cascade levels using the deterministic flow sampler. Since the child is conditioned on finer-scale image features, the refinement recovers new surface detail while preserving global structure inherited from the parent. The parameter t_s controls the trade-off between preserving the parent geometry and incorporating finer image details available at

the child level. As $t_s \to 0$, the child remains close to the parent. As $t_s \to 1$, it relies mainly on local high-resolution image features and loses the global context that the parent provides. The global context helps resolve ambiguities that cannot be determined from a single cell alone. To balance local detail and global context, we set $t_s = 0.8$ empirically for all benchmarks, without tuning per scene, dataset, or view count.

The parent also determines the child-cell layout by retaining only cells that intersect its predicted geometry, restricting generation to relevant regions. To initialize the cells of the coarsest level and per-cell view selection, we use the point cloud from Depth Anything 3 (DA3) (Lin et al., 2026). We use the depth only for these two choices, and never fuse it into the output geometry (Appendix D.3).

3.3 Multi-Scale Training of a Shared Model

Since the parent prediction is used only to initialize the next level at inference time via Equation 1, training does not require paired coarse and fine examples. We train one SS flow and one mesh flow on individual cells, using standard conditional flow matching (Lipman et al., 2023; Liu et al., 2023). We map cells of several edge lengths s to the same canonical coordinates and R^3 grid and condition the model on the images that observe each cell. Cells at different scales therefore differ only in voxel spacing $\Delta = s/R$. Neither s nor Δ is given to the model, so the same weights apply across scales. Unlike XCube and SCube (Ren et al., 2024a;b), our hierarchy is not fixed during training, so its depth, cell layout, and physical scales can be chosen at inference time without retraining. Appendix E provides more details about training.

3.4 Cell-Independent Multi-Scale Image Conditioning

In the MultiDiffusion formulation above, every cell is evaluated at every flow step of the SS and mesh stages, making feature reuse important for efficient inference. Pixel-aligned conditioning (Li et al., 2026a; Schmid et al., 2026) projects each voxel into the input views, bilinearly samples one feature per view, and aggregates them before the 3D generator. However, the DINOv3 token grid is coarse, so the sampled features can lose the information needed for detailed surface reconstruction. Pixal3D (Li et al., 2026a) addresses this by upsampling features to image resolution, but the upsampled maps become costly to cache as image resolution and view count increase, and must then be recomputed at every flow step. Aggregating before the generator also keeps the view weights from depending on the current voxel state. We address these with voxel-queried image attention over compact cached features and a cell-independent tile bank that provides each cell with scale-matched image features without per-cell encoding.

Voxel-queried image attention. For a voxel p, we gather the DINO tokens within a $w \times w$ window around its projection in each view, and let the current voxel feature h_p attend to all of them with a single softmax over tokens and views. Instead of bilinear sampling, we apply relative positional encoding to the continuous offset from the voxel projection to the token center to preserve sub-token position and local image context. As the query is the current voxel feature, the attention weights adapt to the evolving 3D state across transformer blocks and flow steps. We also encode the projected voxel size and ray direction (Li et al., 2025) (Appendix D.1). For finer detail, the mesh stage also samples compact multi-stride ConvNeXt features (Liu et al., 2022) at the same projected location. The DINO features remain on their native token grid, while the ConvNeXt features are lightweight, allowing both to be cached for all views and reused across cells and flow steps.

Cell-independent tile bank. The base generator (Xiang et al., 2026) is pretrained on 512×512 images that frame the target region at a particular image scale. To match this pretraining setup, each view should provide a crop that covers the cell projection, with pixel spacing matched to the cell's voxel spacing. GenRecon (Schmid et al., 2026) instead resizes each full image to a single resolution. In wide scenes, where an image covers a large area, this can leave a fine cell with only a few pixels. Instead of center-cropping the images around each cell projection, which would require separate image encoding for every cell–view pair and dominate the cost, we build an isotropic pyramid for each input image and divide each level into a fixed grid of 512×512 tiles (Figure 2(b)). Each tile is defined by its source image, pyramid level, and crop origin, independently of the 3D cells. For each view, a cell selects the level that matches its voxel spacing and the tiles covering its projection, so fine cells use the original image while coarser cells use downsampled levels. We treat each tile as a virtual posed view by incorporating its scale and crop origin into the intrinsics, which keeps the



Figure 3: **Qualitative results.** (a) ScanNet++ `7b6477cb95` and (b) Tanks and Temples `truck`. (c) Real-world captures (ours): mesh, one input view, and a close-up of the boxed region.

input format of pretraining. At each level, we encode only the selected tile union and reuse it across cells, the SS and mesh stages, and all flow steps. The tile bank also benefits training, where the same tiles are reused across cell scales and 3D layouts. Appendices D and G give details and ablations.

4 Experiments

4.1 Experimental Setup

Models and training data. We train two models (Appendix E): a general scene reconstruction model and a city-block reconstruction model. We initialize both from a pretrained TRELLIS.2 (Xiang et al., 2026) model, first fine-tune it for multi-view conditioning on TexVerse (Zhang et al., 2025), and then fine-tune the two models independently on their respective datasets. The general model is trained exclusively on synthetic data: the single-room datasets SAGE-10K (Xia et al., 2026) and 3D-FRONT (Fu et al., 2021), together with our procedurally generated multi-room houses and grass scenes. The city-block model is fine-tuned on our dataset of photographs captured in Japanese cities and their paired building information models (BIMs). The exterior geometry of the BIMs defines the training target and contains no people or vehicles (Figure 4; Appendix B.1).

Inference. In all experiments, each cell has an SS grid with $R = 64$, and the mesh decoder produces meshes at 512^3 resolution. We apply margin-based chunk decoding to obtain a complete scene-wide mesh (Appendix F.1). We place the cells at a stride of $s/2$ at every level. We use 12 flow-matching steps in the SS and mesh stages, with classifier-free guidance (Ho & Salimans, 2022) scales of 7.5

Table 1: **Quantitative comparison on ScanNet++.** Mean over the 25 largest scenes. 2D metrics compare depth and normals rendered at held-out views, and 3D metrics compare the mesh surfaces visible from those views. L0 to L2 are our cascade levels. The top three results are highlighted.

	32 Views							256 Views						
	2D				3D			2D				3D		
Method	MAE↓	AbsRel↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	AbsRel↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑
w/ per-scene optimization														
2DGS	0.1426	0.1141	38.376	0.9058	0.0723	0.4837	0.7181	0.0929	0.0846	32.208	0.9747	0.0513	0.5345	0.8006
MILo	0.2523	0.1780	42.156	0.9848	0.1127	0.4241	0.6910	0.1406	0.1226	33.612	0.9998	0.0817	0.5508	0.7985
GaussianWrapping	0.3467	0.2762	35.974	0.9648	0.2830	0.4413	0.7505	0.3197	0.3071	30.118	0.9761	0.2136	0.6075	0.8113
w/o per-scene optimization														
DA3	0.0811	0.0800	28.000	0.9682	0.0212	0.7765	0.8658	0.0818	0.0792	20.789	0.9958	0.0193	0.7882	0.9128
MapAnything	0.1019	0.1213	30.090	0.9275	0.0249	0.6716	0.8458	0.0907	0.0969	23.982	0.9905	0.0255	0.5761	0.8916
Murre	0.1990	0.2041	40.515	0.9431	0.0638	0.3867	0.7180	0.2759	0.2792	40.780	0.9803	0.0916	0.3471	0.6952
GenRecon	0.1175	0.1228	20.928	0.9764	0.0604	0.2846	0.8652	0.1049	0.1099	20.030	0.9782	0.0611	0.2834	0.8723
Ours (L0)	0.0714	0.0746	19.207	0.9984	0.0235	0.6336	0.9090	0.0683	0.0677	18.952	0.9990	0.0233	0.6251	0.9125
Ours (L1)	0.0569	0.0575	17.098	0.9983	0.0199	0.6959	0.9232	0.0571	0.0535	16.684	0.9892	0.0217	0.6874	0.9224
Ours (L2)	0.0565	0.0536	16.330	0.9962	0.0212	0.7155	0.9247	0.0557	0.0499	15.624	0.9902	0.0202	0.7176	0.9283

and 4.0, respectively. At finer levels, the SS stage starts from the parent at $t_s = 0.8$ and runs $K = 10$ steps. Each cell uses at most 64 conditioning image tiles. We use three cascade levels, numbered from level 0 (coarsest) to level 2 (finest), and four on the city blocks.

Datasets. We evaluate on ScanNet++ (Yeshwanth et al., 2023), six scenes from Tanks and Temples (Knapitsch et al., 2017), and our city-block dataset. For ScanNet++, we use the 25 validation scenes with the largest ground-truth bounding-box volumes, with 32 or 256 input views selected by farthest-point sampling. For Tanks and Temples, we use 16 or 256 input views of the 2K images. Appendix A.2 provides results with 32 views and 4K inputs. We use all views for scenes with fewer than 256. For the city blocks, we use three held-out blocks with 256 input views. We also show `garden` from Mip-NeRF 360 (Barron et al., 2022) (Figure 1), and `coffee_martini` from N3DV (Li et al., 2022) and an in-house capture of five people (Figure 3(c)).

Baselines. Our baselines include per-scene optimization methods (2DGS (Huang et al., 2024), MILo (Guédon et al., 2025), GaussianWrapping (Gomez et al., 2026), and CityGaussianV2 (Liu et al., 2025)), feed-forward methods (Depth Anything 3 (DA3) (Lin et al., 2026), MapAnything (Keetha et al., 2026), and Murre (Guo et al., 2025)), and the generative method GenRecon (Schmid et al., 2026). We evaluate all but CityGaussianV2 on ScanNet++, also drop GenRecon on Tanks and Temples, and use 2DGS, CityGaussianV2, DA3, and Murre on the city blocks. All but MILo, GaussianWrapping, and GenRecon are meshed by a shared TSDF fusion (Appendix B.4).

Metrics. We report Chamfer distance (CD) in meters, F-score, and normal consistency (NC). F@τ denotes the F-score at a threshold of τ cm. We report F@2 on ScanNet++ and F@10 on the city blocks. For Tanks and Temples, we use the official per-scene threshold τ, which ranges from 3 to 25 mm. Because scene scale varies widely across its scenes, we also divide CD by τ before averaging and report CD/τ. ScanNet++ and Tanks and Temples are evaluated using the virtual-scan protocol of GaussianWrapping (Gomez et al., 2026), which scores only surfaces seen from the cameras of the capture. For ScanNet++, we also report the held-out-view 2D metrics used by GenRecon (Schmid et al., 2026), including depth MAE and AbsRel, normal angular error, and completeness. For the city blocks, the BIM meshes model surfaces unseen from the street, such as roofs, so we evaluate whole blocks, cropping only the prediction to the ground-truth footprint. See Appendix B for details.

4.2 Comparison with Prior Reconstruction Methods

ScanNet++. Table 1 reports results on the 25 largest ScanNet++ scenes with 32 and 256 input views. Our finest level outperforms GenRecon, the closest generative baseline, on all seven metrics in both settings. DA3 obtains a higher F@2 in both settings and a slightly lower Chamfer distance and higher completeness with 256 views, but its depth and normal errors are larger than ours. In Figure 3(a), DA3 produces bumpy surfaces and misses small objects on the desk, while our method recovers flat surfaces, the keyboard, and pens. Our accuracy changes little between 32 and 256 views, whereas all optimization-based methods degrade with 32 views (e.g., F@2 of 2DGS drops from 0.53 to 0.48). Without per-scene optimization, our method outperforms all optimization-based baselines on all three 3D metrics. Appendix A.1 gives further results on all 50 validation scenes.

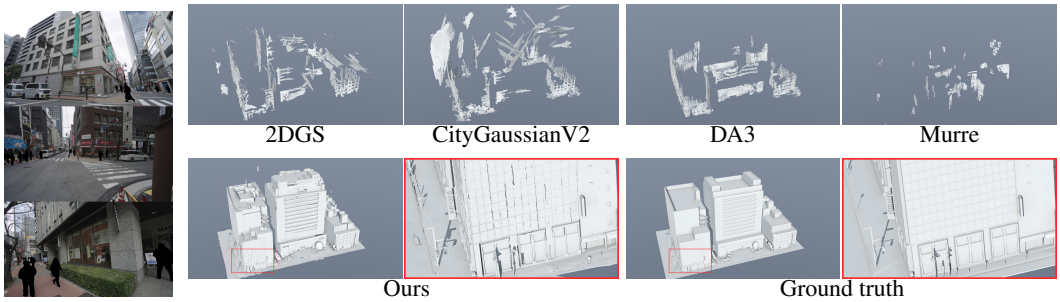
Table 2: **Quantitative comparison on Tanks and Temples and the city-block dataset.** (a) Mean over the three small (Caterpillar, Ignatius, Truck) and three large (Barn, Courthouse, Meetingroom) scenes with 2K input images. F@τ and CD/τ use the per-scene threshold τ. (b) Mean over three blocks, evaluated against the BIM ground truth. Lx denotes level x, with its cell size in parentheses. The top three results are highlighted.

	Small, 16 Views		Small, 256 Views		Large, 16 Views		Large, 256 Views	
Method	F@$\tau\uparrow$	CD/$\tau\downarrow$	F@$\tau\uparrow$	CD/$\tau\downarrow$	F@$\tau\uparrow$	CD/$\tau\downarrow$	F@$\tau\uparrow$	CD/$\tau\downarrow$
w/ per-scene optimization								
2DGS	0.355	9.42	0.448	9.64	0.101	44.95	0.233	28.08
MILo	0.346	14.79	0.527	11.63	0.053	54.04	0.268	51.29
GaussianWrapping	0.345	8.80	0.650	5.54	0.094	37.61	0.370	42.51
w/o per-scene optimization								
DA3	0.148	12.48	0.199	9.25	0.114	35.34	0.180	24.26
MapAnything	0.071	19.23	0.077	27.90	0.014	194.69	0.039	86.44
Murre	0.051	47.77	0.058	31.85	0.051	42.13	0.060	49.72
Ours (L0)	0.219	8.19	0.249	8.04	0.175	15.71	0.232	7.08
Ours (L1)	0.278	7.57	0.313	7.29	0.195	12.45	0.240	6.90
Ours (L2)	0.299	7.28	0.354	6.95	0.189	11.11	0.248	7.00

(a) Tanks and Temples

Method	CD ↓	F@10 ↑	NC ↑
w/ per-scene optimization			
2DGS	4.858	0.076	0.174
CityGaussianV2	5.155	0.112	0.228
w/o per-scene optimization			
DA3	5.184	0.015	0.048
Murre	9.998	0.001	0.010
Ours (L0, 80 m)	0.872	0.245	0.269
Ours (L1, 40 m)	0.714	0.296	0.315
Ours (L2, 20 m)	0.662	0.351	0.350
Ours (L3, 10 m)	0.645	0.367	0.361

(b) City blocks, 256 input views



(a) Input views (b) Reconstruction results

Figure 4: **Qualitative comparison on the city-block dataset.** (a) Three of the 256 input views. (b) Each method's reconstruction, with zoomed crops of ours and the ground truth to compare detail.

Tanks and Temples. Table 2(a) reports the mean over three small and three large scenes. Optimization-based methods achieve higher F-scores with dense observations. Still, our method achieves the lowest or a competitive Chamfer distance in all settings and the highest F-score on the large scenes with 16 views. Since our method does not fit the input images pixel by pixel, its surfaces can be slightly offset from the ground truth, which the tight thresholds of 3 to 25 mm penalize. Our meshes are nevertheless more complete and detailed (Figures 3(b) and 16). None of the baselines reconstructs the transparent, reflective windows of `truck` as closed surfaces, while our generative prior completes them. Appendix A.2 gives per-scene results, results with 4K input images, and further qualitative comparisons.

City blocks. Table 2(b) reports results on the three held-out blocks with 256 input views. These scenes are challenging because the street-level photographs contain depths from a few meters to over a hundred meters. All baselines fail in this setting, with Chamfer distances of about 5 m or more, whereas our method reaches 0.65 m and outperforms them on all three metrics. Each finer cascade level improves all three metrics, and Figure 4 shows the reconstructed blocks with close-ups.

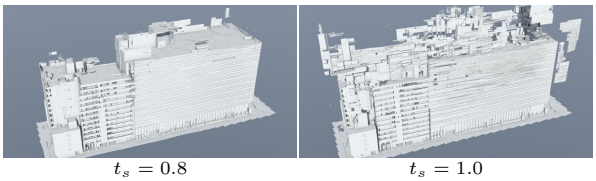
Real-world captures. Figure 1 shows `garden`, where finer levels around the table recover the plants in the vase, the gaps between the table planks, the metal plate at its center, and a glossy sphere in the background. Appendix A.4 compares this scene with GaussianWrapping. In the other two captures (Figure 3(c)), we recover transparent glass bottles, a badge lanyard, and shirt wrinkles.

4.3 Ablations and Analysis

Parent initialization. Each finer level starts its flow sampling from the parent's prediction noised to t_s instead of pure noise (Section 3.2). We compare our default $t_s = 0.8$ with $t_s = 0.9$ and $t_s = 1$ at every finer level (Figure 5(a)). At $t_s = 1$, the finer levels start from pure noise and inherit only the

	2D		3D	
t_s	MAE ↓	Ang ↓	CD ↓	F@2 ↑
0.8	**0.0565**	**16.33**	**0.0212**	**0.7155**
0.9	0.1450	19.16	0.0621	0.6779
1.0	0.2170	22.15	0.1051	0.6306

(a) ScanNet++, 32 input views



(b) City block, 256 input views

Figure 5: **Effect of parent initialization.** Every finer cascade level starts from a latent noised to t_s, where $t_s = 1$ generates from pure noise. (a) Mean over the 25 largest ScanNet++ scenes, with the best results in bold. (b) One block of the city-block dataset.

	2D		3D	
Conditioning	MAE ↓	Ang ↓	CD ↓	F@2 ↑
Baseline	0.0610	16.44	0.0228	0.6424
Voxel attn.	0.0609	16.34	0.0235	0.6495
+ ConvNeXt	**0.0557**	**15.62**	**0.0202**	**0.7176**

(a) ScanNet++, 256 input views



(b) Human capture, 56 input views

Figure 6: **Effect of image conditioning.** The baseline uses the GenRecon-style conditioner, our voxel-queried attention replaces it, and the ConvNeXt branch is then added, all at the finest cascade level. (a) Mean over the 25 largest ScanNet++ scenes, with the best results in bold. (b) A multi-view capture of people. Faces are obscured for anonymous review.

cell placement. On the large ScanNet++ scenes with 32 views, lowering t_s improves every metric and reduces the Chamfer distance by 80%. Results with 256 views are in Appendix G.1. Figure 5(b) shows a city block. Overlapping cells keep neighboring surfaces continuous, but information from distant cells barely reaches a cell, and the roof is not visible from the street. Without the parent, the roof breaks into fragments above the building. With the parent, all cells start from the same coarse building, and the roof stays in place.

Image conditioning. Figure 6 compares, at the finest level, the GenRecon-style conditioner, our voxel-queried attention (Section 3.4), and the latter with the ConvNeXt branch, which adds pixel-level detail. The baseline bilinearly samples DINO features at projected voxel centers, which loses detail finer than a token. On ScanNet++, voxel-queried attention alone is on par with the baseline, and the ConvNeXt branch substantially improves every metric. On the human capture, the baseline over-smooths the hair, the nose, and the shirt wrinkles. Voxel-queried attention recovers strands of hair, and the ConvNeXt branch sharpens them further and adds the shirt buttons. Appendix G.2 shows further qualitative results.

Inference time. On large ScanNet++ scenes, a single H200 takes 17 (L1) and 89 (L2) minutes per scene, comparable to per-scene optimization but slower than feed-forward methods (Appendix H).

5 Conclusion and Limitations

We have presented T3lescope, a generative reconstruction framework that applies a single fixed-resolution 3D generator across scene scales through an inference-time coarse-to-fine cascade. The coarse level establishes a globally consistent layout, finer levels recover surface detail, and a cell-independent multi-scale image conditioning gives every cell images at the resolution it needs. On ScanNet++, Tanks and Temples, and city blocks, T3lescope outperforms the generative baseline and is competitive with or better than feed-forward and per-scene optimization methods.

Since T3lescope does not fit the input images pixel by pixel, its surfaces can be slightly offset from the observations, which lowers the F-score at tight thresholds when views are dense. Its cost grows with the number of cells and cascade levels, so sampling large scenes at high resolution remains slow, and it assumes posed images. Moreover, it does not work on distant objects in unbounded scenes (Appendix A.5). In future work, we plan to reduce the sampling cost by distilling the flow models into few-step samplers and to extend T3lescope to unbounded scenes.

AI USE STATEMENT

In this work, we used generative AI tools to search for related work, to propose and refine hypotheses, to give feedback on the design of the method and the experiments, and to implement the methods, including the model, training, inference, and evaluation code. We also used them to generate the synthetic training datasets, namely the procedural houses and grass scenes (Appendix E.3), to clean and reformat datasets, and to interpret the results, including quantitative analyses of the experimental outputs. Additionally, we used generative AI tools to draft and edit the paper for readability and to create figures. We have reviewed all AI-assisted work. The authors specified the implementation, checked the implemented code against these specifications, checked the qualitative and quantitative results, and reviewed the AI-assisted text. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our city-block images are taken on public streets and show passers-by and vehicles. In the released data, people and license plates are masked, and the released 3D Gaussian Splatting models and BIM meshes contain no people (Appendix B.1). The five people in our in-house human capture consented to being recorded. All other datasets are public and used under their own terms.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices C and D describe the method, including the inference procedure, the cell sizes and extents, and the image conditioning. Appendix E gives the training data, the curriculum, and the optimization settings, including the number of steps and GPUs, and Appendix B gives the datasets, metrics, evaluation protocols, and baseline settings. We will release the code upon acceptance. All datasets other than ours are public, and Appendix B.1 describes which parts of our city-block data we will release.

REFERENCES

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *International Conference on Machine Learning (ICML)*, 2023.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5470–5479, 2022.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. PGSR: Planar-based Gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 31(9):6100–6111, 2025.

Ruoyi Du, Dongliang Chang, Timothy Hospedales, Yi-Zhe Song, and Zhanyu Ma. DemoFusion: Democratising high-resolution image generation with no $$$. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3d furnished rooms with layouts and semantics. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Diego Gomez, Antoine Guédon, Nissim Maruani, Bingchen Gong, and Maks Ovsjanikov. From blobs to spokes: High-fidelity surface reconstruction via oriented Gaussians. In *European Conference on Computer Vision (ECCV)*, pp. 54–72, 2026.

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 (TOG)*, 44(6):1–15, 2025.

Haoyu Guo, He Zhu, Sida Peng, Haotong Lin, Yunzhi Yan, Tao Xie, Wenguan Wang, Xiaowei Zhou, and Hujun Bao. Multi-view reconstruction via SfM-guided monocular depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research (JMLR)*, 23(47):1–33, 2022.

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

Amogh Joshi, Julian Ost, and Felix Heide. WorldFlow3D: Flowing through 3D distributions for unbounded world generation. In *European Conference on Computer Vision (ECCV)*, pp. 517–534, 2026.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, et al. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*, 2026.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4), 2017. doi: 10.1145/3072959.3073599.

Dong-Yang Li, Wang Zhao, Yuxin Chen, Wenbo Hu, Meng-Hao Guo, Fang-Lue Zhang, Ying Shan, and Shi-Min Hu. Pixal3D: Pixel-aligned 3D generation from images. In *ACM SIGGRAPH Conference Papers*, 2026a.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Sikuang Li, Chen Yang, Jiemin Fang, Taoran Yi, Jia Lu, Jiazhong Cen, Lingxi Xie, Wei Shen, and Qi Tian. WorldGrow: Generating infinite 3D world. In *AAAI Conference on Artificial Intelligence*, 2026b.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3D video synthesis from multi-view video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, 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.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Yang Liu, Chuanchen Luo, Zhongkai Mao, Junran Peng, and Zhaoxiang Zhang. CityGaussianV2: Efficient and geometrically accurate reconstruction for large-scale scenes. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, 2022.

Yifan Lu, Xuanchi Ren, Jiawei Yang, Tianchang Shen, Zhangjie Wu, Jun Gao, Yue Wang, Siheng Chen, Mike Chen, Sanja Fidler, and Jiahui Huang. InfiniCube: Unbounded and controllable dynamic 3D driving scene generation with world-guided video models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. LT3SD: Latent trees for 3D scene diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Quan Meng, Yujin Chen, Lei Li, Matthias Nießner, and Angela Dai. Seen2Scene: Completing realistic 3D scenes with visibility-guided flow. In *European Conference on Computer Vision (ECCV)*, pp. 386–404, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. XCube: Large-scale 3D generative modeling using sparse voxel hierarchies. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

Xuanchi Ren, Yifan Lu, Hanxue Liang, Zhangjie Wu, Huan Ling, Mike Chen, Sanja Fidler, Francis Williams, and Jiahui Huang. SCube: Instant large-scale scene reconstruction using VoxSplats. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Katharina Schmid, Nicolas von Lützow, Jozef Hladký, Angela Dai, and Matthias Nießner. GenRecon: Bridging generative priors for multi-view 3D scene reconstruction. *arXiv preprint arXiv:2605.23888*, 2026.

Johannes Lutz Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, pp. 501–518, 2016.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Frédo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

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 (CVPR)*, 2024.

Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. BlockFusion: Expandable 3D scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable agentic 3d scene generation for embodied ai. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22358–22368, 2026.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3d indoor scenes. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. TexVerse: A universe of 3D objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018.

A ADDITIONAL RESULTS

This section gives the complete results on each benchmark of the main paper and qualitative comparisons on captures outside the benchmarks. The metrics and evaluation protocols are defined in Appendix B.

A.1 SCANNET++

Tables 3 and 4 report results on all 50 validation scenes with 32 and 256 input views, split into the 25 smallest and the 25 largest by ground-truth bounding-box volume. The large split is the one in Table 1. Table 4 also reports the whole-mesh protocol of GenRecon. DA3 obtains the lowest MAE, AbsRel, and Chamfer distance on the small scenes, while our finest level has the lowest normal error in all four settings. Figures 7 and 8 show four scenes with 32 input views, and Figures 9 and 10 with 256. Figure 11 compares close-ups of our meshes with those of GenRecon on `0d2ee665be` and `578511c8a9` in the 256-view setting.

Table 3: **Quantitative comparison on ScanNet++ (2D metrics).** Mean over the 25 smallest, the 25 largest, and all 50 scenes. Depth and normals are rendered from the meshes at held-out views. The top three results are highlighted.

	Small (25)						Large (25)						All (50)					
	Depth				Normal	Coverage	Depth				Normal	Coverage	Depth				Normal	Coverage
Method	MAE↓	RMSE↓	AbsRel↓	SqRel↓	Ang↓	Comp↑	MAE↓	RMSE↓	AbsRel↓	SqRel↓	Ang↓	Comp↑	MAE↓	RMSE↓	AbsRel↓	SqRel↓	Ang↓	Comp↑
32 Views																		
2DGS	0.1261	0.2493	0.1329	0.0591	40.149	0.9542	0.1426	0.3412	0.1141	0.0799	38.376	0.9058	0.1344	0.2953	0.1235	0.0695	39.262	0.9300
MILo	0.1748	0.3699	0.1855	0.1530	44.731	0.9913	0.2523	0.6171	0.1780	0.2623	42.156	0.9848	0.2136	0.4935	0.1817	0.2077	43.444	0.9881
GaussianWrapping	0.4688	1.4378	0.6228	6.9407	40.150	0.9365	0.3467	1.3934	0.2762	2.8323	35.974	0.9648	0.4077	1.4156	0.4495	4.8865	38.062	0.9507
DA3	0.0255	0.0925	0.0348	0.0139	19.773	0.9888	0.0811	0.2849	0.0800	0.0900	28.000	0.9682	0.0533	0.1887	0.0574	0.0520	23.886	0.9785
MapAnything	0.0335	0.1143	0.0508	0.0249	20.971	0.9717	0.1019	0.3277	0.1213	0.1466	30.090	0.9275	0.0677	0.2210	0.0860	0.0857	25.530	0.9496
Murre	0.0758	0.1779	0.1024	0.0501	32.075	0.9816	0.1990	0.4693	0.2041	0.2290	40.515	0.9431	0.1374	0.3236	0.1532	0.1396	36.295	0.9623
GenRecon	0.0571	0.1311	0.0739	0.0224	18.123	0.9983	0.1175	0.2939	0.1228	0.1004	20.928	0.9764	0.0873	0.2125	0.0984	0.0614	19.526	0.9873
Ours (L0)	0.0438	0.1157	0.0585	0.0176	18.745	0.9986	0.0714	0.2394	0.0746	0.0599	19.207	0.9984	0.0576	0.1775	0.0666	0.0387	18.976	0.9985
Ours (L1)	0.0346	0.0924	0.0440	0.0096	16.351	0.9994	0.0569	0.2046	0.0575	0.0431	17.098	0.9983	0.0457	0.1485	0.0508	0.0264	16.724	0.9988
Ours (L2)	0.0324	0.0913	0.0407	0.0090	15.312	0.9990	0.0565	0.2132	0.0536	0.0408	16.330	0.9962	0.0445	0.1522	0.0471	0.0249	15.821	0.9976
256 Views																		
2DGS	0.0962	0.1901	0.1117	0.0386	37.382	0.9794	0.0929	0.2391	0.0846	0.0537	32.208	0.9747	0.0946	0.2146	0.0981	0.0462	34.795	0.9771
MILo	0.1535	0.5276	0.1718	0.5368	39.667	0.9978	0.1406	0.5426	0.1226	0.3784	33.612	0.9998	0.1470	0.5351	0.1472	0.4576	36.639	0.9988
GaussianWrapping	0.4076	1.2380	0.5049	3.5345	35.282	0.9321	0.3197	1.4556	0.3071	5.7070	30.118	0.9761	0.3636	1.3468	0.4060	4.3026	32.700	0.9541
DA3	0.0254	0.0928	0.0348	0.0140	16.238	0.9957	0.0818	0.2910	0.0792	0.0966	20.789	0.9958	0.0536	0.1919	0.0570	0.0553	18.513	0.9958
MapAnything	0.0345	0.1080	0.0480	0.0187	18.299	0.9922	0.0907	0.2944	0.0969	0.1038	23.982	0.9905	0.0626	0.2012	0.0724	0.0613	21.140	0.9914
Murre	0.0962	0.2185	0.1236	0.0713	29.964	0.9926	0.2759	0.5960	0.2792	0.3498	40.780	0.9803	0.1861	0.4073	0.2014	0.2106	35.372	0.9864
GenRecon	0.0577	0.1337	0.0742	0.0237	17.690	0.9981	0.1049	0.2732	0.1099	0.0867	20.030	0.9782	0.0813	0.2035	0.0921	0.0552	18.860	0.9881
Ours (L0)	0.0469	0.1167	0.0603	0.0155	18.980	0.9988	0.0683	0.2358	0.0677	0.0533	18.952	0.9990	0.0576	0.1762	0.0640	0.0344	18.966	0.9989
Ours (L1)	0.0375	0.0954	0.0478	0.0101	16.440	0.9995	0.0571	0.2102	0.0535	0.0426	16.684	0.9892	0.0473	0.1528	0.0506	0.0264	16.562	0.9943
Ours (L2)	0.0361	0.0963	0.0446	0.0110	15.493	0.9991	0.0557	0.2131	0.0499	0.0418	15.624	0.9902	0.0459	0.1547	0.0472	0.0264	15.558	0.9947

Table 4: **Quantitative comparison on ScanNet++ (3D metrics).** Mean over the 25 smallest, the 25 largest, and all 50 scenes. We report the whole-mesh protocol of GenRecon (Schmid et al., 2026) and the virtual-scan protocol of GaussianWrapping (Gomez et al., 2026). The top three results are highlighted.

	Small (25)							Large (25)							All (50)					
	Whole mesh			Virtual scan				Whole mesh			Virtual scan				Whole mesh			Virtual scan		
Method	CD↓	F@10↑	NC↑	CD↓	F@2↑	F@10↑	NC↑	CD↓	F@10↑	NC↑	CD↓	F@2↑	F@10↑	NC↑	CD↓	F@10↑	NC↑	CD↓	F@2↑	F@10↑	NC↑
32 Views																					
2DGS	0.0682	0.7878	0.6587	0.0741	0.4453	0.7698	0.7065	0.1017	0.7748	0.6399	0.0723	0.4837	0.8070	0.7181	0.0850	0.7813	0.6493	0.0732	0.4645	0.7884	0.7123
MILo	0.0605	0.8043	0.6696	0.1013	0.4264	0.7381	0.6779	0.0910	0.7708	0.6532	0.1127	0.4241	0.7382	0.6910	0.0757	0.7876	0.6614	0.1070	0.4252	0.7382	0.6845
GaussianWrapping	0.0633	0.8085	0.6728	0.4135	0.4433	0.7501	0.6723	0.0766	0.7909	0.6963	0.2830	0.4413	0.8073	0.7505	0.0699	0.7997	0.6846	0.3483	0.4423	0.7787	0.7114
DA3	0.0166	0.9775	0.8675	0.0113	0.8829	0.9915	0.9200	0.0455	0.9452	0.7846	0.0212	0.7765	0.9773	0.8658	0.0311	0.9614	0.8260	0.0162	0.8297	0.9844	0.8929
MapAnything	0.0186	0.9777	0.8594	0.0132	0.8350	0.9927	0.9104	0.0482	0.9225	0.7607	0.0249	0.6716	0.9662	0.8458	0.0334	0.9501	0.8101	0.0191	0.7533	0.9794	0.8781
Murre	0.0368	0.9186	0.7665	0.0377	0.5269	0.9135	0.8137	0.0699	0.8409	0.6822	0.0638	0.3867	0.8258	0.7180	0.0534	0.8798	0.7244	0.0507	0.4568	0.8697	0.7659
GenRecon	0.0434	0.8736	0.8289	0.0343	0.3885	0.9574	0.8984	0.1007	0.8405	0.7863	0.0604	0.2846	0.8823	0.8652	0.0721	0.8570	0.8076	0.0473	0.3366	0.9198	0.8818
Ours (L0)	0.0275	0.9485	0.8134	0.0230	0.5816	0.9848	0.9110	0.0440	0.9347	0.8112	0.0235	0.6336	0.9768	0.9090	0.0358	0.9416	0.8123	0.0233	0.6076	0.9808	0.9100
Ours (L1)	0.0259	0.9494	0.8295	0.0200	0.6359	0.9915	0.9240	0.0433	0.9361	0.8284	0.0199	0.6959	0.9830	0.9232	0.0346	0.9427	0.8289	0.0199	0.6659	0.9873	0.9236
Ours (L2)	0.0251	0.9502	0.8371	0.0190	0.6718	0.9914	0.9296	0.0439	0.9329	0.8360	0.0212	0.7155	0.9805	0.9247	0.0345	0.9416	0.8365	0.0201	0.6936	0.9859	0.9272
256 Views																					
2DGS	0.0583	0.8216	0.6711	0.0631	0.4776	0.8021	0.7468	0.0762	0.8463	0.6962	0.0513	0.5345	0.8668	0.8006	0.0672	0.8340	0.6836	0.0572	0.5060	0.8345	0.7737
MILo	0.0511	0.8423	0.7070	0.1292	0.4963	0.8140	0.7491	0.0611	0.8428	0.7260	0.0817	0.5508	0.8508	0.7985	0.0561	0.8425	0.7165	0.1054	0.5235	0.8324	0.7738
GaussianWrapping	0.0496	0.8490	0.7273	0.3414	0.5675	0.7924	0.7324	0.0607	0.8511	0.7563	0.2136	0.6075	0.8562	0.8113	0.0552	0.8501	0.7418	0.2775	0.5875	0.8243	0.7718
DA3	0.0167	0.9779	0.8911	0.0119	0.8784	0.9898	0.9368	0.0381	0.9591	0.8438	0.0193	0.7882	0.9825	0.9128	0.0274	0.9685	0.8675	0.0156	0.8333	0.9861	0.9248
MapAnything	0.0209	0.9785	0.8784	0.0165	0.7386	0.9924	0.9246	0.0417	0.9460	0.8181	0.0255	0.5761	0.9746	0.8916	0.0313	0.9623	0.8482	0.0210	0.6573	0.9835	0.9081
Murre	0.0420	0.8778	0.7754	0.0434	0.5441	0.8909	0.8301	0.0776	0.7827	0.6667	0.0916	0.3471	0.7447	0.6952	0.0598	0.8302	0.7210	0.0675	0.4456	0.8178	0.7626
GenRecon	0.0427	0.8822	0.8303	0.0343	0.3628	0.9680	0.8997	0.0975	0.8602	0.7982	0.0611	0.2834	0.8997	0.8723	0.0701	0.8712	0.8142	0.0477	0.3231	0.9339	0.8860
Ours (L0)	0.0287	0.9519	0.8093	0.0279	0.4897	0.9777	0.9086	0.0382	0.9392	0.8136	0.0233	0.6251	0.9787	0.9125	0.0334	0.9456	0.8114	0.0256	0.5574	0.9782	0.9106
Ours (L1)	0.0268	0.9533	0.8233	0.0238	0.5726	0.9859	0.9211	0.0402	0.9388	0.8304	0.0217	0.6874	0.9811	0.9224	0.0335	0.9460	0.8268	0.0227	0.6300	0.9835	0.9217
Ours (L2)	0.0260	0.9533	0.8324	0.0234	0.6167	0.9843	0.9260	0.0401	0.9377	0.8419	0.0202	0.7176	0.9821	0.9283	0.0330	0.9455	0.8372	0.0218	0.6672	0.9832	0.9271



Figure 7: **Qualitative comparison on ScanNet++ with 32 input views.** All meshes are rendered from a held-out view of each scene.

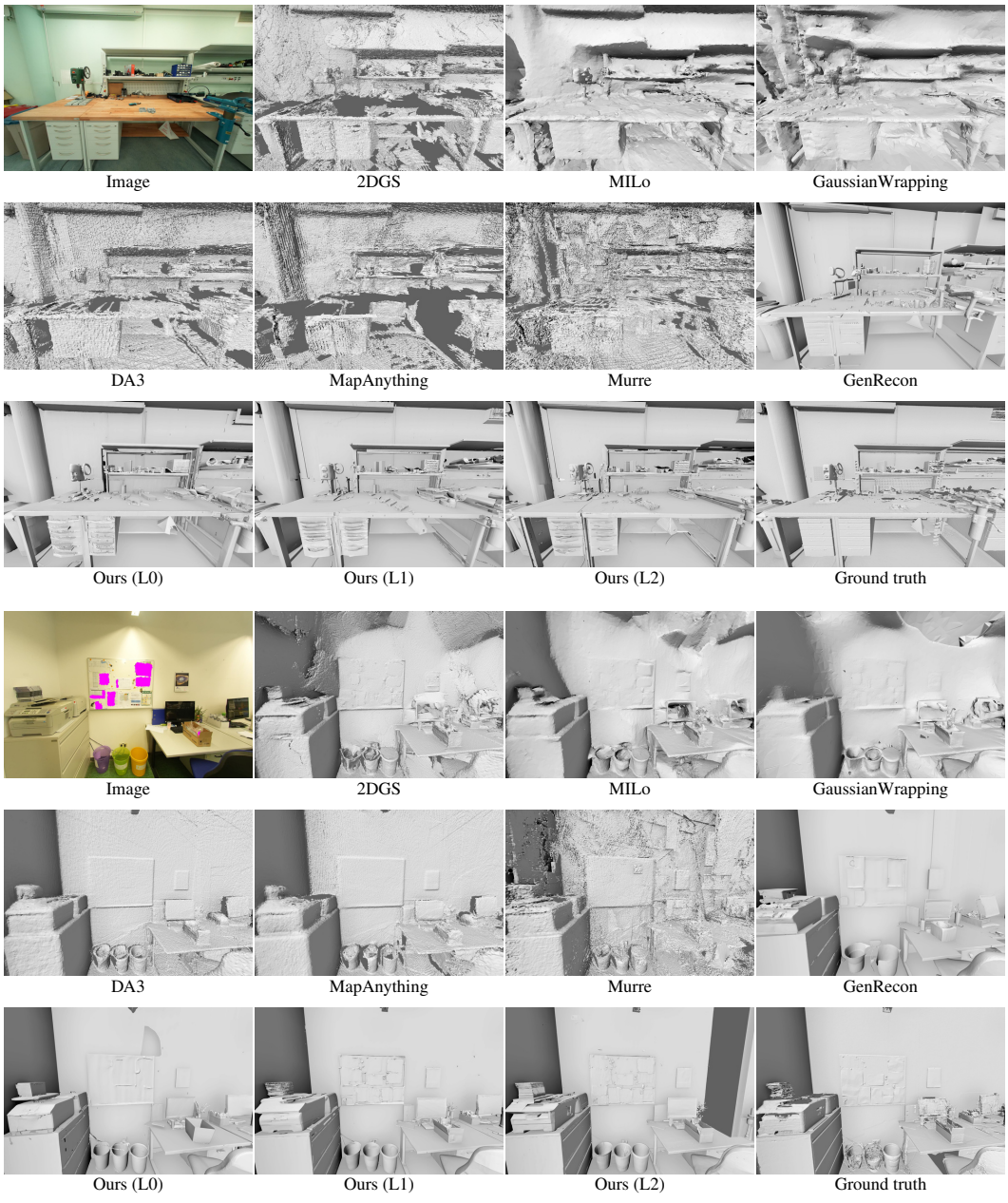


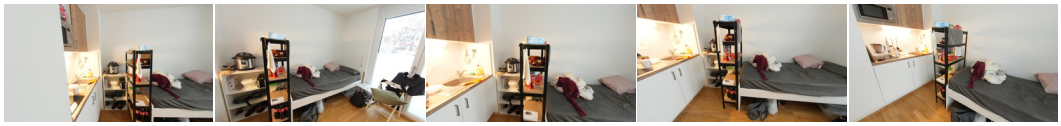
Figure 8: **Qualitative comparison on ScanNet++ with 32 input views.** All meshes are rendered from a held-out view of each scene.



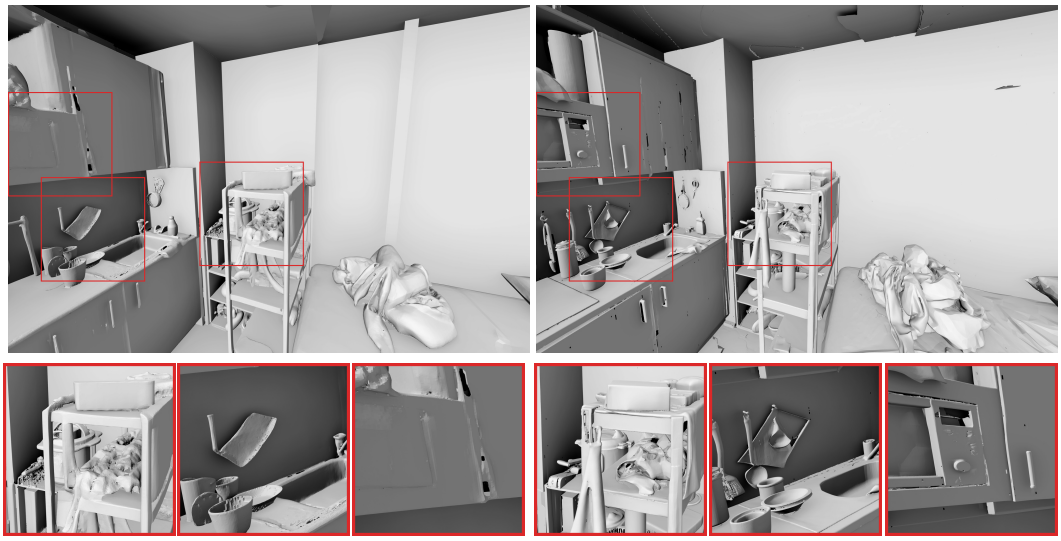
Figure 9: **Qualitative comparison on ScanNet++ with 256 input views.** All meshes are rendered from a held-out view of each scene.



Figure 10: **Qualitative comparison on ScanNet++ with 256 input views.** All meshes are rendered from a held-out view of each scene.



0d2ee665be (ScanNet++) – five of the 79 input views at 1752 × 1168 resolution



GenRecon Ours



578511c8a9 (ScanNet++) – five of the 253 input views at 1752 × 1168 resolution



GenRecon Ours

Figure 11: **Qualitative comparison with GenRecon on ScanNet++ in the 256-view setting.** Both meshes are rendered from the same held-out viewpoint, with close-ups of the boxed regions below.

A.2 TANKS AND TEMPLES

Table 5 reports per-scene results with 16, 32, and 256 input views. Since each scene is listed separately, the Chamfer distance is given in meters rather than divided by τ. The rows marked 4K use the 4K images instead of the 2K ones (Appendix B.1). With the 4K images, the finest level reaches a higher mean F@τ in every view setting (e.g., 0.3344 vs. 0.3009 with 256 views), and with 256 views also a lower CD/τ. The gain is concentrated at the finer levels, because only their cells read the full-resolution tiles. Figures 12 and 13 show all six scenes with 16 input views, and Figures 14 and 15 with 256. Figure 16 compares close-ups of our meshes with those of GaussianWrapping, the per-scene optimization baseline with the highest F@τ, on `ignatius` and `courthouse` in the 256-view setting.

Table 5: **Per-scene quantitative comparison on Tanks and Temples.** F@τ uses the official per-scene threshold τ, and CD is in meters. Rows marked 4K use the 4K input images instead of the 2K ones. The top three results are highlighted.

	Barn		Caterpillar		Courthouse		Ignatius		Meetingroom		Truck	
Method	F@$\tau\uparrow$	CD$\downarrow$	F@$\tau\uparrow$	CD$\downarrow$	F@$\tau\uparrow$	CD$\downarrow$	F@$\tau\uparrow$	CD$\downarrow$	F@$\tau\uparrow$	CD$\downarrow$	F@$\tau\uparrow$	CD$\downarrow$
16 Views												
2DGS	0.1680	0.4530	0.2467	0.0667	0.0773	1.1360	0.3975	0.0260	0.0566	0.4412	0.4213	0.0314
MILo	0.0622	0.5128	0.1522	0.1275	0.0235	2.3359	0.5351	0.0210	0.0740	0.1741	0.3510	0.0594
GaussianWrapping	0.1520	0.1269	0.2098	0.0777	0.0457	2.0951	0.5061	0.0081	0.0840	0.1635	0.3192	0.0409
DA3	0.1123	0.2928	0.1088	0.0876	0.0418	1.3350	0.1417	0.0349	0.1878	0.2334	0.1926	0.0414
MapAnything	0.0055	2.9563	0.0445	0.1260	0.0010	6.2998	0.0870	0.0451	0.0352	0.3645	0.0815	0.0873
Murre	0.0652	0.2628	0.0283	0.2581	0.0467	1.4180	0.0263	0.2148	0.0408	0.4338	0.0998	0.1005
Ours (L0)	0.1655	0.0600	0.1230	0.0767	0.0869	0.8989	0.2633	0.0091	0.2729	0.0519	0.2699	0.0311
Ours (L1)	0.1985	0.0764	0.1569	0.0710	0.1057	0.6151	0.3453	0.0070	0.2797	0.0513	0.3316	0.0309
Ours (L2)	0.1926	0.0789	0.1631	0.0679	0.1079	0.4984	0.3898	0.0067	0.2675	0.0553	0.3430	0.0302
Ours (L0), 4K input	0.1894	0.0630	0.1140	0.0773	0.0874	0.8770	0.2691	0.0090	0.2958	0.0508	0.2577	0.0307
Ours (L1), 4K input	0.2081	0.0523	0.1711	0.0669	0.1087	0.7036	0.3987	0.0063	0.2965	0.0484	0.3221	0.0308
Ours (L2), 4K input	0.2080	0.0895	0.1949	0.0637	0.1159	0.5905	0.4704	0.0055	0.3084	0.0476	0.3563	0.0311
32 Views												
2DGS	0.2741	0.2013	0.2771	0.0526	0.0998	1.1360	0.6163	0.0074	0.1083	0.3033	0.4419	0.0292
MILo	0.2216	0.6706	0.2154	0.1273	0.0459	1.8242	0.6513	0.0478	0.1363	0.1274	0.4545	0.0447
GaussianWrapping	0.2633	0.0688	0.2882	0.0732	0.1167	1.6180	0.6007	0.0050	0.1309	0.1154	0.3909	0.0375
DA3	0.1371	0.2588	0.1387	0.0650	0.0394	1.0687	0.1809	0.0162	0.2832	0.1259	0.2244	0.0382
MapAnything	0.0117	0.7994	0.0672	0.0842	0.0021	5.4052	0.1000	0.0271	0.0647	0.2432	0.1072	0.0718
Murre	0.0637	0.2332	0.0295	0.2111	0.0513	1.1645	0.0199	0.1984	0.0664	0.2941	0.1085	0.0810
Ours (L0)	0.1854	0.0671	0.1367	0.0781	0.1008	0.3191	0.3211	0.0077	0.3402	0.0344	0.3010	0.0322
Ours (L1)	0.2113	0.0528	0.1795	0.0713	0.1190	0.2639	0.3897	0.0069	0.3491	0.0332	0.3542	0.0310
Ours (L2)	0.2069	0.0778	0.1890	0.0672	0.1231	0.2298	0.4244	0.0062	0.3396	0.0343	0.3751	0.0306
Ours (L0), 4K input	0.1463	0.0787	0.1328	0.0805	0.0965	0.3619	0.3223	0.0073	0.3561	0.0349	0.2955	0.0317
Ours (L1), 4K input	0.1926	0.0641	0.1752	0.0727	0.1181	0.3355	0.4194	0.0058	0.3721	0.0318	0.3523	0.0305
Ours (L2), 4K input	0.1938	0.0866	0.1945	0.0740	0.1324	0.2553	0.4904	0.0051	0.3806	0.0308	0.3705	0.0304
256 Views												
2DGS	0.4156	0.0602	0.2566	0.0725	0.1000	1.6849	0.6432	0.0232	0.1841	0.1082	0.4427	0.0334
MILo	0.4617	0.7530	0.3048	0.1064	0.1172	1.6766	0.7452	0.0193	0.2250	0.1151	0.5314	0.0359
GaussianWrapping	0.6366	0.0285	0.4850	0.0549	0.1796	2.9165	0.8413	0.0022	0.2945	0.0802	0.6226	0.0245
DA3	0.1256	0.3737	0.1320	0.0750	0.0634	0.7696	0.2420	0.0096	0.3521	0.0462	0.2232	0.0478
MapAnything	0.0080	1.0894	0.0303	0.2213	0.0043	3.2754	0.0959	0.0263	0.1045	0.1936	0.1033	0.1533
Murre	0.0503	0.3124	0.0391	0.2477	0.0308	2.5481	0.0410	0.0896	0.0977	0.1601	0.0941	0.0806
Ours (L0)	0.2004	0.0672	0.1305	0.0775	0.1077	0.3000	0.3083	0.0076	0.3885	0.0253	0.3081	0.0305
Ours (L1)	0.2070	0.0624	0.1846	0.0694	0.1283	0.2926	0.4109	0.0058	0.3833	0.0277	0.3419	0.0304
Ours (L2)	0.2238	0.0601	0.2036	0.0654	0.1352	0.3052	0.4739	0.0053	0.3839	0.0280	0.3849	0.0302
Ours (L0), 4K input	0.2303	0.0576	0.1347	0.0773	0.1127	0.2764	0.3178	0.0077	0.3832	0.0258	0.2813	0.0316
Ours (L1), 4K input	0.2570	0.0514	0.1867	0.0672	0.1292	0.2547	0.4331	0.0056	0.4049	0.0294	0.3601	0.0305
Ours (L2), 4K input	0.2791	0.0516	0.2243	0.0632	0.1472	0.2377	0.5365	0.0046	0.4185	0.0269	0.4008	0.0299



Figure 12: **Qualitative comparison on Tanks and Temples with 16 input views.** All meshes of a scene are rendered from the same viewpoint.

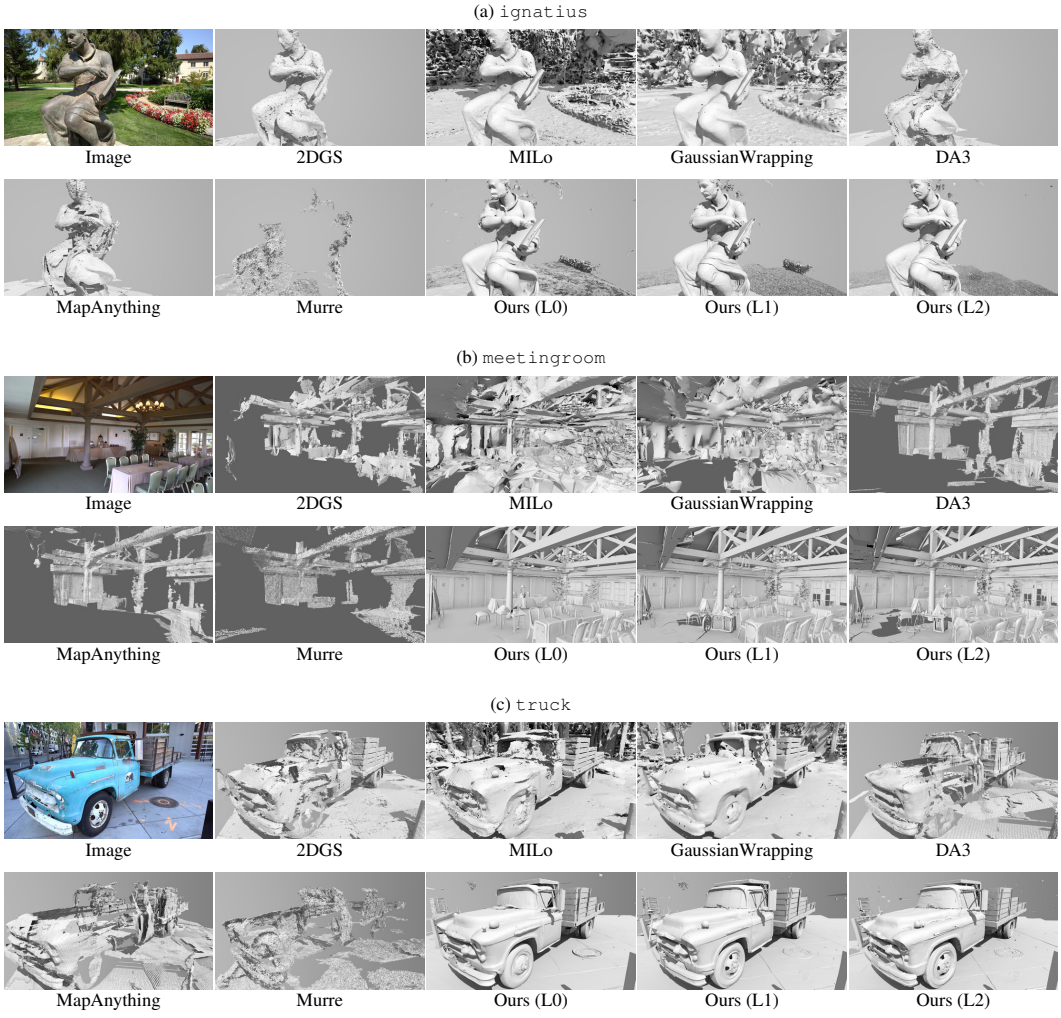


Figure 13: **Qualitative comparison on Tanks and Temples with 16 input views.** All meshes of a scene are rendered from the same viewpoint.

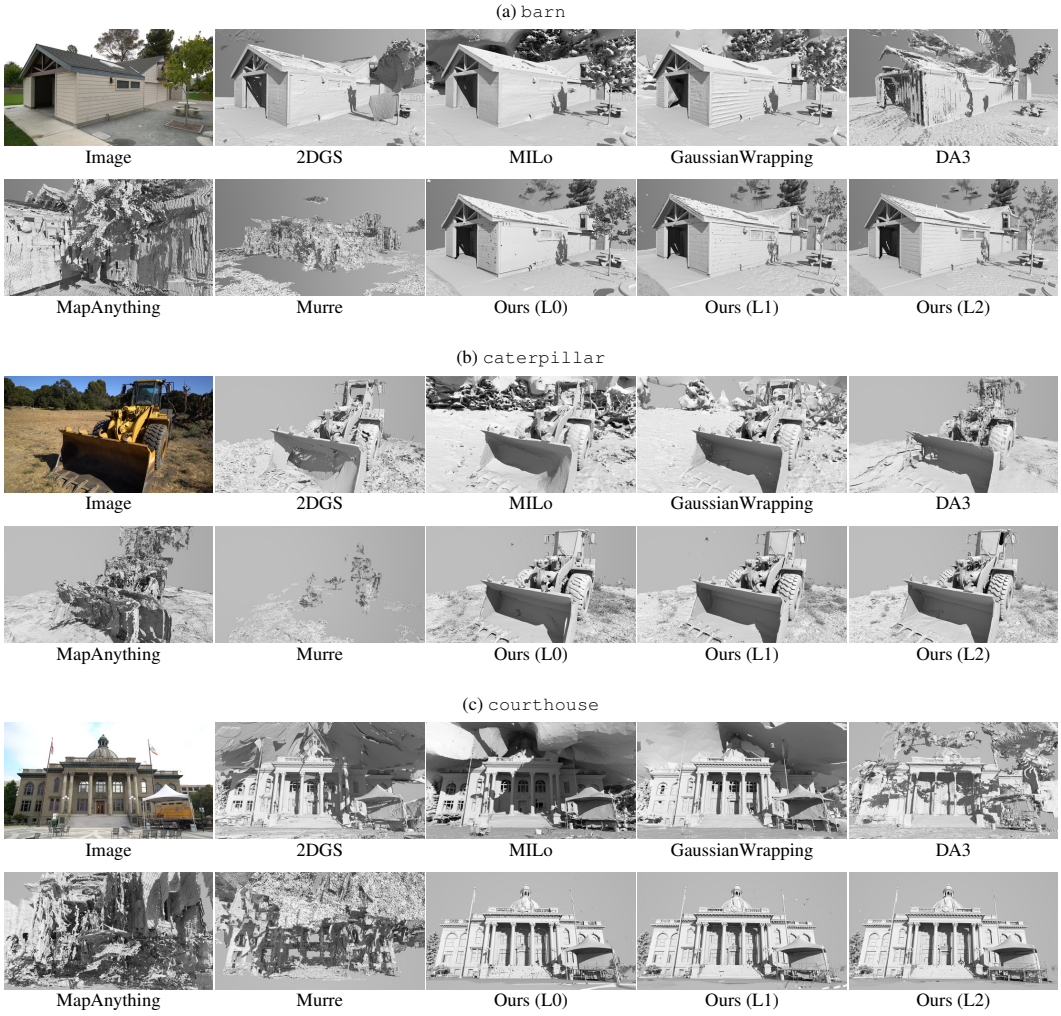


Figure 14: **Qualitative comparison on Tanks and Temples with 256 input views.** All meshes of a scene are rendered from the same viewpoint.

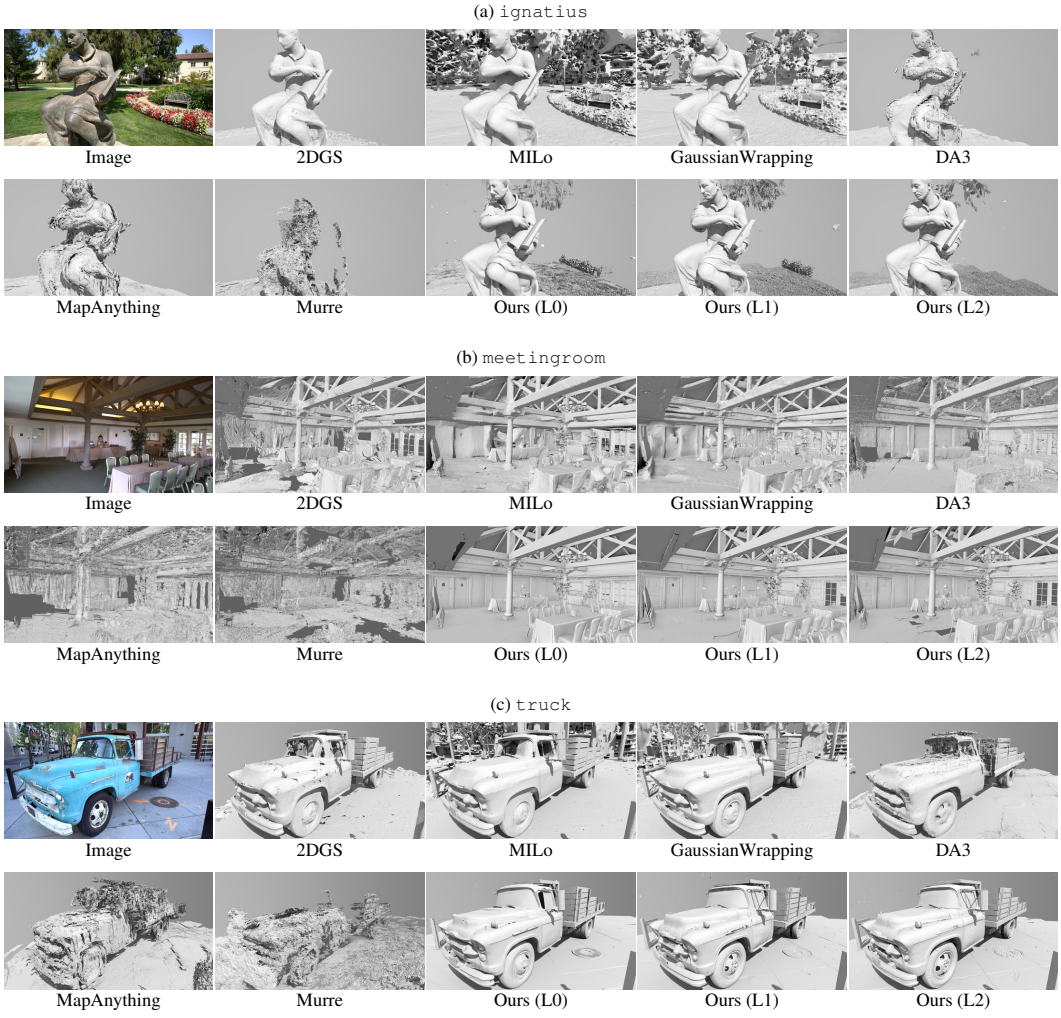


Figure 15: **Qualitative comparison on Tanks and Temples with 256 input views.** All meshes of a scene are rendered from the same viewpoint.



Figure 16: **Qualitative comparison with GaussianWrapping on Tanks and Temples with 256 input views.** Both meshes are rendered from the same viewpoint, with close-ups of the boxed regions below.

A.3 CITY BLOCKS

Table 6 reports per-block results with 256 input views. Each finer cascade level improves all three metrics on every block.

Table 6: **Per-block quantitative comparison on the city-block dataset.** All methods use 256 input views. The cell size of each level of our method is shown in parentheses. CD is in meters. The top three results are highlighted.

	Scene 025			Scene 033			Scene 034			Mean		
Method	CD ↓	F@10 ↑	NC ↑	CD ↓	F@10 ↑	NC ↑	CD ↓	F@10 ↑	NC ↑	CD ↓	F@10 ↑	NC ↑
w/ per-scene optimization												
2DGS	4.701	0.101	0.231	5.011	0.088	0.157	4.863	0.039	0.134	4.858	0.076	0.174
CityGaussianV2	6.278	0.119	0.216	4.262	0.123	0.217	4.926	0.094	0.252	5.155	0.112	0.228
w/o per-scene optimization												
DA3	4.514	0.014	0.039	4.754	0.010	0.027	6.285	0.022	0.079	5.184	0.015	0.048
Murre	10.744	0.001	0.008	10.189	0.001	0.021	9.061	0.000	0.002	9.998	0.001	0.010
Ours (L0, 80 m)	0.967	0.226	0.239	0.795	0.202	0.251	0.855	0.307	0.317	0.872	0.245	0.269
Ours (L1, 40 m)	0.792	0.282	0.297	0.673	0.272	0.304	0.676	0.335	0.345	0.714	0.296	0.315
Ours (L2, 20 m)	0.775	0.323	0.330	0.603	0.325	0.337	0.607	0.404	0.382	0.662	0.351	0.350
Ours (L3, 10 m)	0.747	0.339	0.339	0.596	0.338	0.349	0.591	0.423	0.396	0.645	0.367	0.361

A.4 COMPARISON WITH GAUSSIANWRAPPING ON REAL-WORLD CAPTURES

We compare our method with GaussianWrapping, which achieves the highest F@τ among the per-scene optimization baselines on Tanks and Temples with 256 views. Figure 17 shows `garden` from Mip-NeRF 360 (Barron et al., 2022) and an in-house capture of five people. We show the number and resolution of our input views in the figure. Both methods use the same input views, but GaussianWrapping takes them at 2K resolution. Their meshes are rendered from the same camera with the same shading.



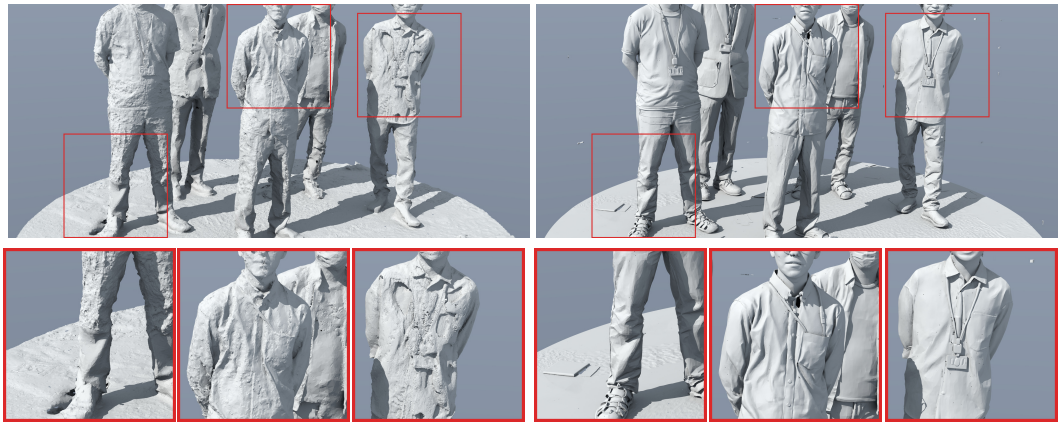
`garden` (Mip-NeRF 360, Barron et al., 2022) – five of the 185 input views at 5187×3361 resolution



GaussianWrapping Ours



in-house person capture – five of the 56 input views at 3840×2160 resolution



GaussianWrapping Ours

Figure 17: **Qualitative comparison with GaussianWrapping on real-world captures.** Both meshes are rendered from the same viewpoint, with close-ups of the boxed regions below. Faces are obscured for anonymous review.

A.5 FAILURE CASE IN UNBOUNDED SCENES

Since our method sets the scale and placement of cells freely, it can in principle reconstruct distant regions by placing large cells far from the cameras. We reconstruct `ignatius` and `caterpillar` from Tanks and Temples with 256 input views using such far and coarse cells. We use four cascade levels and place cells up to eight times farther from the scene center than the cameras. We render the meshes from an input camera and from the top in Figure 18. Despite these far cells, our method does not work well on distant objects such as the ground and the trees behind the central object. This might be caused by the limited diversity of the training data. Moreover, by construction, our method cannot handle objects at infinite distance.

(a) `ignatius`



Input image | Ours | Ours (top view)

(b) `caterpillar`



Input image | Ours | Ours (top view)

Figure 18: **Failure cases with distant objects.** Input images containing distant objects, and our meshes from 256 input views at 4K, rendered from the same cameras and from the top. The distant objects are not reconstructed well.

B Experimental Setup Details

B.1 Datasets

ScanNet++. We evaluate the 50 validation scenes of ScanNet++ (Yeshwanth et al., 2023) and divide them by the volume of the ground-truth bounding box into a small and a large split of 25 scenes each. We use the undistorted DSLR images at 1752×1168 as inputs and choose the 32- and 256-view sets by farthest-point sampling of the camera centers among the training frames not flagged as bad. For the scenes with fewer than 256 frames, 12 in the small split and 9 in the large split, we use all of them. The ground truth is the mesh reconstructed from the laser scans, with its anonymized regions removed.

Tanks and Temples. We evaluate six scenes of Tanks and Temples (Knapitsch et al., 2017). As for ScanNet++, the 16-, 32-, and 256-view sets are chosen by farthest-point sampling of the camera centers. `truck` has only 251 images and uses all of them in the 256-view setting. We extract the frames from the 4K videos provided by the dataset and undistort them to match the official COLMAP reconstruction, which gives images of approximately 3920×2180 (4K). For a fair comparison with the baselines, the main paper uses these images downsampled by two to approximately 1960×1090 (2K), and Appendix A.2 reports additional results with the 4K images. The ground truth is the official laser scan with its crop volume and alignment.

City blocks. Our city-block dataset contains 41 blocks with 248 buildings and 139,893 posed images at 9504×6336. We split it geographically into 38 training blocks with 220 buildings and 126,068 images, and three held-out evaluation blocks with 28 buildings and 13,825 images. The blocks range from 42×22 m to 156×137 m in horizontal extent, and the buildings are 23 to 172 m tall. All images are captured with a handheld camera at walking height, about 1 to 2 m above the ground.

We register the images using RealityScan and an in-house SfM pipeline, and align the reconstruction to a laser scan of the block to obtain metric scale. The exterior BIM of each building is modeled in the coordinate frame of the same scan. The cameras and the BIM are therefore aligned without any further registration. We use the mesh derived from the BIM as the ground truth. It represents building exteriors, vegetation, and street furniture down to signs and traffic cones, but not people or vehicles. Each image is accompanied by masks of the sky, people, and vehicles. We will release the evaluation split, including the images, a 3D Gaussian Splatting model of each block, and the BIM meshes. For evaluation, each held-out block is reconstructed from 256 of its images, sampled at a uniform stride over the capture sequence.

B.2 Metrics

3D metrics. Let P and G denote the points sampled from the predicted mesh and the ground truth, respectively, following each protocol (Appendix B.3). The Chamfer distance (CD) is the average of the mean nearest-neighbor distances from P to G and from G to P, in meters and without truncation. Precision and recall at a threshold τ are the fractions of points in P and in G whose nearest neighbor in the other set lies within τ, and the F-score F@τ is their harmonic mean, with τ in centimeters. Normal consistency (NC) is the mean absolute cosine similarity between the normal of each point in P and G and that of its nearest neighbor in the other set, with pairs more than 20 cm apart counted as zero.

2D metrics. We follow the 2D metrics of GenRecon (Schmid et al., 2026). We render depth and normal maps of the predicted and ground-truth meshes and compute the errors on the pixels covered by both. We report the mean absolute error (MAE) and root mean squared error (RMSE) of depth in meters, the absolute relative error (AbsRel) $\mathrm{mean}\,|\hat{z} - z|/z$, and the squared relative error (SqRel) $\mathrm{mean}(\hat{z} - z)^2/z$, where $\hat{z}$ and z are the predicted and ground-truth depths. Ang is the mean angular error in degrees between the rendered normals, which are interpolated from area-weighted vertex normals. Completeness (Comp) is the fraction of pixels covered by the ground truth that are also covered by the prediction. For depth, the main paper reports MAE and AbsRel, and Table 3 additionally reports RMSE and SqRel.

B.3 Evaluation Protocols

ScanNet++. Although ScanNet++ registers the camera poses to the ground-truth mesh, the registration leaves residual errors on some scenes that are not negligible at the F@2 threshold. We therefore rigidly align the ground-truth mesh to the COLMAP sparse points of each scene with ICP before evaluation. This alignment is shared by all methods.

We compute the 2D metrics following GenRecon (Schmid et al., 2026), rendering both meshes from the cameras of the official test split, invalidating masked pixels, and pooling the errors over all pixels of all frames. As its evaluation code is not released, we reimplement it based on the paper.

For the 3D metrics, the main paper uses the virtual-scan protocol of GaussianWrapping (Gomez et al., 2026), which scores only the surfaces that the cameras see and the ground truth contains. Unlike the original protocol, which uses all cameras of the capture, we use up to 300 held-out cameras chosen by farthest-point sampling among the frames that are neither input views of the run nor flagged as bad. We cast only the rays that hit the ground truth against the prediction, so a missing surface lowers recall and a surface outside the ground truth is never scored.

Table 4 also reports the 3D evaluation protocol of GenRecon, which samples points uniformly over the complete meshes after discarding predicted triangles farther than 15 cm from the ground truth. We run its released code with four million points per mesh instead of 200,000 to reduce sampling noise. Since the ground truth misses windows, mirrors, ceilings, and surfaces behind furniture, this protocol cannot distinguish a hole in the ground truth from a hole in the prediction.

Tanks and Temples. We use the virtual-scan protocol of GaussianWrapping. The protocol takes the points seen from all cameras of the official trajectory, crops them to the official volume, and aligns them to the ground truth by the released transform refined with ICP. F@τ uses the official per-scene threshold τ of 3 mm for `ignatius`, 5 mm for `caterpillar` and `truck`, 10 mm for `barn` and `meetingroom`, and 25 mm for `courthouse`. The main paper reports CD divided by τ to average over scenes, and Table 5 reports the per-scene CD in meters. On both ScanNet++ and Tanks and Temples, all methods are scored by the same implementation. Since our meshes can reach hundreds of millions of triangles, we implement a custom rendering and evaluation script that handles such meshes without decimation.

City blocks. As the BIM ground truth covers complete building exteriors, we compare the complete meshes rather than only the surfaces seen by the cameras. We crop only the reconstructions to the ground-truth footprint, the ground-plane projection of the ground-truth mesh, to remove their surroundings outside the block. Since NC counts only pairs within 20 cm, on this dataset it mainly reflects how much of the surface is recovered.

B.4 Baselines

All baselines receive the same input views, camera poses, and intrinsics, and are evaluated on the meshes they produce. We use the official implementations and checkpoints. On ScanNet++ and Tanks and Temples, our method and the per-scene optimization methods take the input images at full resolution on ScanNet++ and at 2K on Tanks and Temples. The feed-forward methods and GenRecon resize them to the resolution expected by each model. The rows marked 4K in Table 5 instead give our method the higher-resolution images of Tanks and Temples. On city blocks, our method takes the original 9504×6336 images, whereas the baselines take undistorted images downsampled to 1920×1280. All methods are also given the extent of the block from the ground truth.

2DGS (Huang et al., 2024) uses a depth ratio of 1 and a distortion weight of 10 on ScanNet++ and city blocks, and the released settings on Tanks and Temples, with the masks applied to the photometric loss on city blocks. **MILo** (Guédon et al., 2025) uses the released fast configuration, with its preset for indoor scenes on ScanNet++ and the released evaluation settings on Tanks and Temples. **GaussianWrapping** (Gomez et al., 2026) uses the released training and mesh extraction settings. **CityGaussianV2** (Liu et al., 2025) uses the released street-scene configuration, which trains a coarse model and fine-tunes 2×2 partitions before merging them, with depth supervision from Depth Anything V2 (Yang et al., 2024) and the masks.

DA3 (Lin et al., 2026) uses DA3-GIANT-1.1 conditioned on the input poses and intrinsics and processes all views in a single forward pass, so that one scale converts the depth to meters. On ScanNet++, each image is split into four virtual cameras with narrower fields of view. **MapAnything** (Keetha et al., 2026) uses the released model conditioned on the input poses and intrinsics. **Murre** (Guo et al., 2025) uses the released settings, with sparse depth from the COLMAP points, or from the RealityScan export on city blocks.

GenRecon (Schmid et al., 2026) partitions the scene into overlapping chunks placed from the COLMAP sparse points, generates each chunk with a TRELLIS.2-based generative model conditioned on the input images, and aggregates the overlapping latents into a single mesh. We run its released code in the ScanNet++ mode with the default settings, conditioning on all input views.

Mesh extraction. MILo, GaussianWrapping, and GenRecon output meshes directly. The other methods produce depth maps, which we fuse by TSDF fusion in Open3D (Zhou et al., 2018) with the ground-truth poses, keeping the 50 largest connected components. The voxel size, truncation, and maximum depth are 1 cm, 4 cm, and 4.5 m on ScanNet++, following GenRecon, and 5 cm, 8 cm, and 200 m on city blocks. On Tanks and Temples, we use the original per-scene settings of 2DGS. For 2DGS on ScanNet++, we use the voxel size of 0.6 cm and truncation of 2.4 cm that its authors use for large Tanks and Temples scenes. For CityGaussianV2, pixels with opacity below 0.5 are discarded before fusion. On city blocks, we mask the depth and crop the volume to the ground-truth extent enlarged by 10 m. Since Murre aligns each view to its own sparse depth, overlapping views are not made consistent, and on city blocks its fused mesh covers only a small part of the ground truth (Table 6).

C Additional Method Details

The full inference procedure for one scene is as follows. The subsections below describe the generator (Appendix C.1), its training (Appendix C.2), the scene-wide blending (Appendix C.3), the transfer between levels (Appendix C.4), and the cell sizes and extents (Appendix C.5).

1. For each level $l = 0, \dots, L-1$, with cell edge length $s_l = s_0/2^l$:
   (a) Place the cells. At level 0, they cover the scene extent; at finer levels, they cover the surface predicted by the parent (Appendices C.5 and C.4).
   (b) Select the conditioning views and tiles for each cell (Appendix D.3).
   (c) Encode the tiles selected at this level into the tile bank, encoding each tile only once even if selected by multiple cells (Appendix D.2).
   (d) Run the SS stage on the scene-wide SS latent grid with MultiDiffusion, starting from noise at level 0 and from the noised parent latent at finer levels, and decode the predicted occupancy $\widehat{O}^{(l)}$.
   (e) Run the mesh stage on the active voxels of $\widehat{O}^{(l)}$ with MultiDiffusion.
   (f) Unless l is the finest level, decode the shape latent up to resolution $2R$ and encode the blocks of the child cells as their inherited SS latents.
2. Decode the shape latent of the finest level, and optionally of other levels (Appendix F.2), into a mesh with margin-based chunk decoding (Appendix F.1).

C.1 Two-stage 3D flow backbone

Our generator follows the two-stage design of TRELLIS.2 (Xiang et al., 2026). It operates on one cell in canonical cell coordinates (Appendix C.2) and is conditioned on the posed images $\mathcal{I}$ that observe the cell (Section 3.1). The SS stage generates the occupancy $O \in \{0,1\}^{R^3}$ of the cell with $R = 64$, where active voxels ($O = 1$) contain surface geometry. The mesh stage then generates a shape latent on the active voxels, which the shape VAE decoder converts into a triangle mesh at 512^3 resolution. Both stages are latent flow-matching models. The SS latent of an occupancy is $x_0 = A_{\mathrm{SS}}(E_{\mathrm{SS}}(O))$, where E_{SS} is the SS VAE encoder and A_{SS} normalizes latents using the mean and standard deviation of the training latents. The shape latent is obtained from the ground-truth mesh by the shape VAE encoder. We keep both pretrained VAEs frozen (Appendix E.1) and train only the two flow models.

Training. For either stage $q \in \{\mathrm{SS}, \mathrm{mesh}\}$, a clean latent x_0 is moved to time $t \in [0, 1]$ along the linear path

$$x_t = (1 - t)\, x_0 + \sigma_t\, \epsilon, \qquad \sigma_t = \sigma_{\min} + (1 - \sigma_{\min})\, t, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

interpolating between data at $t = 0$ and noise at $t = 1$, with $\sigma_{\min} = 10^{-5}$ (Appendix E.6). The velocity network v_{θ_q} is trained to predict the time derivative of this path with the conditional flow-matching objective

$$\mathcal{L}_{\mathrm{FM}}(\theta_q) = \mathbb{E}_{x_0, t, \epsilon}\left[\left\| v_{\theta_q}(x_t, t; \mathcal{I}) - ((1 - \sigma_{\min})\, \epsilon - x_0)\right\|_2^2\right]. \tag{3}$$

The mesh flow is additionally conditioned on the occupancy O. The two flows are trained separately. Appendix C.2 describes how cells of different sizes are sampled for this objective, and Appendix D describes how $\mathcal{I}$ enters the network.

Inference. Each stage starts from Gaussian noise at $t = 1$ and integrates $\mathrm{d}x_t/\mathrm{d}t = v_{\theta_q}(x_t, t; \mathcal{I})$ to $t = 0$ with Euler steps and classifier-free guidance. The SS stage applies the inverse of A_{SS} to the resulting latent and decodes it with the SS VAE decoder into the predicted occupancy $\widehat{O}$. The mesh stage samples a shape latent on the active voxels of $\widehat{O}$ and decodes it into a mesh. Every cell at every cascade level goes through this sequence with the same SS flow, mesh flow, VAEs, and image conditioner. In the full pipeline, overlapping cells share each Euler step through scene-wide MultiDiffusion (Appendix C.3). At finer levels, the SS stage starts from the noised parent latent at t_s rather than $t = 1$ (Appendix C.4).

C.2 Multi-scale training with shared weights

For a cubic cell with center o_c and edge length s, we map a world-space point p to canonical cell coordinates $(p - o_c)/s$ and transform the ground-truth geometry and camera poses in the same way. We therefore encode every cell at the same SS and shape resolutions regardless of its size, and s changes only the world-space voxel spacing $\Delta = s/R$. Small cells represent fine geometry, whereas large cells cover more context with the same latent budget.

The general-scene model is trained with cell edge lengths from two ladders: $\{1, 2, 4\}$ m and $\{1.5, 3, 6\}$ m (Appendix E.5). To expose the model evenly to different scene scales, we sample the cell size explicitly: for each training example, we first choose a training data source with the sampling share of Appendix E.3, then one of its edge lengths with equal probability, and finally a cell of that size from one of its scenes. This avoids biasing training toward smaller cells, which are more numerous under spatial sampling. Each stage uses a single set of weights shared across all cell sizes. Each training example consists of only one cell; we do not train on parent–child pairs or multi-level trajectories.

C.3 Scene-wide MultiDiffusion

At each cascade level, we maintain a single scene-wide latent grid rather than separate latents per cell. We start sampling from Gaussian noise on this grid and update the grid in place at every flow step; at finer levels, we start the SS grid from the noised SS latent encoded from the parent's subdivision output (Appendix C.4). We place cells on a lattice with a stride of half the cell edge length, giving a 50% overlap between neighbors. At every flow step, each cell predicts the latent inside its window from the current grid, and we update the grid at a voxel p with the weighted average of the predictions of all cells c that contain p,

$$\widehat{z}(p) = \frac{\sum_{c : p \in c} w_c(p)\, \widehat{z}_c(p)}{\sum_{c : p \in c} w_c(p)}, \tag{4}$$

where $\widehat{z}_c(p)$ is the prediction of cell c at p and $w_c(p)$ is its blending weight. We use a separable Hann window,

$$w_c(p) = \prod_{d \in \{x, y, z\}} \sin^2\!\left(\frac{\pi(j_d + 1/2)}{R_z}\right), \tag{5}$$

where R_z is the latent resolution of the flow (16 for the SS flow and 32 for the mesh flow) and $j_d \in \{0, \ldots, R_z - 1\}$ is the voxel index of p along axis d within cell c. The weight peaks at the

cell center and approaches zero at the cell boundary, reducing the influence of predictions near the boundary. With a half-cell stride, the weights of overlapping cells sum to one along each axis. The SS latent grid is dense, whereas the shape latent grid is sparse and holds only the active voxels; we apply the same averaging to both. We apply classifier-free guidance within each cell before blending: each cell combines its conditional and unconditional velocities and takes an Euler step, and we blend the updated latents of the cells with Equation 4. Since all cells start from the same grid, this is equivalent to blending the guided velocities.

We use the overlapping half-stride cells only for blending. For mesh decoding, we use a non-overlapping base lattice to decode each region once without duplicated surfaces (Appendix F.1). Cell ordering, random seeds, and view selection are deterministic within a run.

C.4 Parent-to-child geometry transfer

The shape VAE decoder upsamples a shape latent through subdivision stages that each double the resolution. For the parent level, we run the decoder only up to resolution $2R$. Because a child cell has half the edge length of its parent, it spans exactly an R^3 block of this grid, which we crop without spatial resampling. We encode the block with the SS VAE encoder to obtain the inherited SS latent $\bar{x}_0^{(l+1)}$ of Section 3.2. For a general power-of-two ratio r between the parent and child edge lengths, we stop at the stage of resolution rR instead. Because the child inherits its SS latent directly from the parent's subdivision output rather than from the parent mesh, we extract intermediate meshes only when needed (e.g., when combining multiple levels, Appendix F.2) and skip them otherwise.

Because the SS flow starts from t_s rather than $t = 1$, it requires fewer Euler steps.

The parent also determines child cell placement: we keep a child cell only if it intersects the parent's predicted occupancy $\widehat{O}^{(l)}$, dilated by one parent SS voxel so that regions narrowly missed at the coarse level can still be recovered. Without this guidance, the location of the surface would be unknown, and fine cells would have to fill the entire scene volume at a much higher cost. The DA3 points used to place the coarsest cells cannot take this role at finer levels: DA3 predicts points only for regions observed in the images, so in wide scenes with sparse views it misses surfaces that the coarse level can still generate. Parent-guided placement is used even at $t_s = 1$, where noise replaces the inherited latent.

C.5 Cell sizes and extents

ScanNet++ and the city blocks use metric coordinates: in both, the camera poses are aligned to laser scans (Appendix B.1). On these datasets, cell edge lengths are specified in meters: 4, 2, and 1 m on ScanNet++, and 80, 40, 20, and 10 m on city blocks. The poses for Tanks and Temples and the three real-world captures (`garden` from Mip-NeRF 360, `coffee_martini` from N3DV (Li et al., 2022), and our in-house human capture) come from SfM and have no metric scale. We rescale camera translations so that the median distance from the cameras to their centroid equals a target value: 6 or 8 on Tanks and Temples, 1 on `coffee_martini`, and 4 on `garden` and the human capture. Cell sizes are defined in these normalized units: 4, 2, and 1 units on Tanks and Temples, `garden`, and `coffee_martini`, and 2.8, 1.4, and 0.7 units on the human capture.

The coarsest cells cover a horizontal extent and a height range. The height range is computed from DA3 points for each scene (Appendix D.3) and is capped at 8 and 4 units on `garden` and the human capture, respectively. The horizontal extent is also computed from DA3 points on ScanNet++, `meetingroom` (Tanks and Temples), and `coffee_martini`. City blocks use the footprint of the ground-truth mesh, while the remaining Tanks and Temples scenes and the human capture use the convex hull of the camera trajectory. On `garden`, cascade levels cover nested squares with half-widths of 16, 8, and 4 units around the camera centroid, and their meshes are combined as described in Appendix F.2.

D IMAGE CONDITIONING AND SCENE PREPARATION

D.1 VOXEL-QUERIED IMAGE ATTENTION

This section details the voxel-queried image attention introduced in Section 3.4. It replaces the image cross-attention in every transformer block of both flows. In the mesh flow, each voxel attends to two types of image features: DINOv3 tokens around its projection in each view, and ConvNeXt features sampled at the projection. Below, we first contrast this design with pixel-aligned conditioning, and then describe the DINO branch (projection, geometric encoding, and attention), the ConvNeXt branch, and the variant used in the SS flow.

Difference from pixel-aligned conditioning. Pixel-aligned generators sample one feature per view at the voxel projection and combine views before the 3D network. This design has two limitations in our setting.

The first is resolution. A DINO token covers a 16×16-pixel patch; thus, features sampled from the token grid, even with bilinear interpolation, lack direct information finer than 16 pixels, which is insufficient for reconstructing fine mesh details. Pixal3D (Li et al., 2026a) upsamples DINO features to image resolution before sampling. For a 512×512 tile, the upsampled 1,024-channel BF16 feature map requires

$$512^2 \cdot 1024 \cdot 2 \text{ bytes} = 0.5 \text{ GiB per view}, \tag{6}$$

and 64 views require 32 GiB. MultiDiffusion evaluates every cell at every flow step, so features exceeding GPU memory must be recomputed at every evaluation.

The second is view aggregation. Pixal3D averages sampled features over views, and GenRecon (Schmid et al., 2026) replaces the average with a learned aggregator conditioned solely on image features. In both approaches, the view weights are independent of the voxel feature.

Our attention addresses both limitations. For resolution, the attention score receives the offset from the voxel projection to each token center, preserving the sub-token position without upsampling. Shallow ConvNeXt feature maps provide image detail finer than 16 pixels at about 1% of the memory of upsampled DINO features. Both features are small enough to keep in memory for all views, so we compute them once and reuse them across cells and flow steps. For aggregation, we keep the tokens of all views separate until the attention step and use the current voxel feature as the query, so that the view weights depend on the voxel feature. The attention score also incorporates the apparent voxel size and the viewing ray.

Projection and token gathering. Let $p \in [-\frac{1}{2}, \frac{1}{2}]^3$ be a voxel center in cell coordinates, and let $h_p \in \mathbb{R}^C$ be its feature after pre-attention layer normalization. View i has rotation R_i, translation t_i, and intrinsics K_i in cell coordinates, where K_i includes the scale and crop origin of the tile (Appendix D.2). The voxel projects to

$$(x_i, y_i, z_i)^\top = R_i p + t_i, \qquad (u_i, v_i)^\top = \pi_i(p) = \begin{pmatrix} f_{x,i} x_i / z_i + c_{x,i} \\ f_{y,i} y_i / z_i + c_{y,i} \end{pmatrix} \tag{7}$$

in pixel coordinates of the 512×512 tile. A view is invalid for p if $z_i \le 0$ or $\pi_i(p)$ lies outside the tile. In each valid view, we gather the $w \times w$ DINOv3 tokens $\mathcal{G}_i(p)$ whose centers (u_{ir}, v_{ir}) are nearest to $\pi_i(p)$. We use $w = 2$ (Appendix G.4), which selects the four tokens around the projection. Each of the $H = 12$ heads computes

$$q_p = \mathrm{RMSNorm}(W_q h_p), \qquad k_{ir} = \mathrm{RMSNorm}(W_k f_{ir}), \qquad v_{ir} = W_v f_{ir}, \tag{8}$$

with head dimension $d = C/H = 128$, where $f_{ir} \in \mathbb{R}^{1024}$ is the DINOv3 feature of token r in view i.

Relative geometric encoding. We aim to run a single model across arbitrary cell layouts, where cells of different sizes can be placed anywhere, cameras can be at very different distances and directions, and output resolution is chosen at inference. The attention should therefore receive identical geometric inputs whenever a cell appears the same in the images, regardless of where it is placed

or how large it is. Standard camera encodings, such as camera-origin coordinates or Plücker coordinates (Sitzmann et al., 2021; Zhang et al., 2024), are measured from the origin of a coordinate frame: their values grow when a camera is far from the origin or looks at a region far from it, so two neighboring cells that see the same image region receive different encodings. We therefore encode only relative quantities. The attention score receives three such inputs: (i) the viewing ray direction of each token in the cell frame, (ii) the pixel offset from the voxel projection to each token center, and (iii) the apparent voxel size in the image, in pixels. Because these represent local directions or pixel lengths, they depend only on how the cell appears in the images, without referencing absolute cell positions, cell indices, cascade levels, or output resolution. Changing the cell layout or enlarging the scene therefore changes none of the inputs beyond what the images themselves show, allowing the same attention mechanism to operate across all scales.

Each quantity is encoded with RoPE (Su et al., 2024) on its own subspace of the head. The head is split into subspaces for the ray direction, horizontal offset, vertical offset, and scale, plus a remainder without encoding. Their dimensions are

$$d_{\mathrm{ray}} = 3\left\lfloor \frac{d}{6} \right\rfloor, \qquad d_{xy} = 2\left\lfloor \frac{d - d_{\mathrm{ray}}}{6} \right\rfloor, \qquad d_{\rho} = 2\left\lfloor \frac{d - d_{\mathrm{ray}} - 2d_{xy}}{2} \right\rfloor, \tag{9}$$

which for $d = 128$ gives 63, 20, 20, 24, and 1. A subspace of dimension d_b uses the RoPE frequencies $\omega_m = 10{,}000^{-m/(d_b/2)}$ for $m = 0, \ldots, d_b/2 - 1$.

(i) Ray direction. In cell coordinates, the viewing ray through the center of token r is

$$d_{ir} = \mathrm{normalize}\big(R_i^\top K_i^{-1}[u_{ir}, v_{ir}, 1]^\top\big)\,. \tag{10}$$

As in PRoPE (Li et al., 2025), which encodes cameras as relative positional encodings in multi-view attention, camera geometry enters the score as a rotation of the key. Let Q_{ir} be the rotation that takes the canonical optical axis to d_{ir}, applied to each of the 21 three-dimensional blocks of the ray subspace of k_{ir}. The voxel query has no camera and is not rotated. The score therefore depends on the direction of the token ray in cell coordinates.

(ii) Image offset. The query is assigned the voxel projection (u_i, v_i) as its position, and the key is assigned the token center (u_{ir}, v_{ir}). With RoPE on each image axis, the score depends only on the offset

$$\delta u_{ir} = u_i - u_{ir}, \qquad \delta v_{ir} = v_i - v_{ir} \tag{11}$$

in pixels. The attention score thus captures where the voxel projects relative to each token center at sub-token precision. With $w = 2$, the four gathered tokens correspond to those used in bilinear interpolation. Whereas bilinear sampling applies fixed weights based purely on offset, our attention weights depend jointly on offsets and features, varying across heads, blocks, and flow steps.

(iii) Projected scale. Let s_{vox} be the voxel width in cell coordinates, $1/32$ in the mesh flow and $1/16$ in the SS flow. The apparent voxel size in view i, as the logarithm of its width in pixels, is

$$\rho_i(p) = \log\left(\frac{s_{\mathrm{vox}}\sqrt{f_{x,i} f_{y,i}}}{z_i}\right). \tag{12}$$

The query is assigned position $\rho_i(p)$ and the key is assigned zero, making the score sensitive to apparent voxel scale, which varies with cascade level and camera distance.

Attention over tokens and views. The three encodings rotate the query and the key. Let $\Theta(x)$ denote the RoPE rotation of a subspace for a scalar position x. With the subspaces ordered as ray, horizontal offset, vertical offset, scale, and remainder, the encoded query and key are

$$\hat{q}_{pi} = \mathrm{diag}\big(I,\ \Theta(u_i),\ \Theta(v_i),\ \Theta(\rho_i(p)),\ 1\big)\, q_p, \qquad \hat{k}_{ir} = \mathrm{diag}\big(Q_{ir},\ \Theta(u_{ir}),\ \Theta(v_{ir}),\ I,\ 1\big)\, k_{ir}. \tag{13}$$

The query depends on the view through the projection and the apparent size, and the key depends only on the token. The attention weights are a single softmax over the gathered tokens of all valid views, so the views are weighted against each other rather than averaged,

$$a_{pir} = \frac{\exp\left(\hat{q}_{pi}^\top \hat{k}_{ir}/\sqrt{d}\right)}{\displaystyle\sum_j \sum_{s \in \mathcal{G}_j(p)} \exp\left(\hat{q}_{pj}^\top \hat{k}_{js}/\sqrt{d}\right)}. \tag{14}$$

The output of the head is $o_p = \sum_i \sum_{r \in \mathcal{G}_i(p)} a_{pir} v_{ir}$.

ConvNeXt detail branch. The DINO tokens contain no direct information finer than 16 pixels. The mesh flow therefore also uses the stride-4, 8, and 16 feature maps of a frozen ConvNeXt-T (Liu et al., 2022), with 96, 192, and 384 channels. Per view, they take

$$2\left(128^2 \cdot 96 + 64^2 \cdot 192 + 32^2 \cdot 384\right) \text{ bytes} \simeq 5.25 \text{ MiB}, \tag{15}$$

or 0.33 GiB for 64 views, compared with 0.5 GiB per view for the upsampled DINO features (Equation 6). The ConvNeXt features of all views are therefore kept in the feature bank and reused across cells and flow steps.

In this branch, we sample each feature map bilinearly at the voxel projection. Let $F_i^{(s)}$ be the stride-s feature map of view i, and let P_s be a linear projection to 96 channels shared by all blocks. The feature of view i is

$$g_{pi} = \underset{s\in\{4,8,16\}}{\text{Concat}} P_s \,\text{Bilinear}\Big(F_i^{(s)}, \pi_i(p)/s\Big) \in \mathbb{R}^{288}. \tag{16}$$

The branch uses four heads of dimension $d_\mathrm{d} = 64$,

$$q_p^\mathrm{d} = \mathrm{RMSNorm}(W_q^\mathrm{d} h_p), \qquad k_{pi}^\mathrm{d} = \mathrm{RMSNorm}(W_k^\mathrm{d} g_{pi}), \qquad v_{pi}^\mathrm{d} = W_v^\mathrm{d} g_{pi}. \tag{17}$$

Because the feature is sampled at the projection, there is no offset to encode, and the head has only the ray and scale subspaces. With Q_{pi} the ray rotation for the ray through $\pi_i(p)$, the encoded query and key are $\hat{q}_{pi}^\mathrm{d} = \mathrm{diag}(I, \Theta(\rho_i(p)))\, q_p^\mathrm{d}$ and $\hat{k}_{pi}^\mathrm{d} = \mathrm{diag}(Q_{pi}, I)\, k_{pi}^\mathrm{d}$. The weights over views and the output are

$$b_{pi} = \underset{i\in\mathcal{V}(p)}{\mathrm{softmax}}\left(\hat{q}_{pi}^{\mathrm{d}\top} \hat{k}_{pi}^\mathrm{d}/\sqrt{d_\mathrm{d}}\right), \qquad o_p^\mathrm{d} = \sum_{i\in\mathcal{V}(p)} b_{pi} v_{pi}^\mathrm{d}, \tag{18}$$

where $\mathcal{V}(p)$ is the set of valid views. The DINO branch thus attends over tokens and views jointly, and the ConvNeXt branch attends over views only.

Both outputs are added to the voxel feature x_p before the layer normalization,

$$\widetilde{x}_p = x_p + \gamma_\mathrm{out} \odot \left(W_o\, o_p + \gamma_\mathrm{d} \odot W_o^\mathrm{d} o_p^\mathrm{d}\right), \tag{19}$$

where o_p and o_p^d concatenate the heads, and γ_out and γ_d are learned per-channel gates initialized to 1 and 0.1.

Attention in the SS flow. We use the attention described above in the mesh flow and a simplified version in the SS flow. The SS flow follows the conditioning of Pixal3D (Li et al., 2026a) and adds the ray and scale encodings to it. Each voxel of the dense 16^3 latent attends to the CLS and register tokens of all views. It also receives the DINO patch features sampled bilinearly at its projection in each view, at their native 32×32 resolution. These sampled features are combined over views by attention with the voxel feature as the query, using the ray and scale encodings of (i) and (iii).

D.2 Multi-scale tile feature bank

Each cell needs images at a pixel spacing matched to its voxel spacing (Section 3.4). Cropping and encoding a separate image for each cell would repeatedly invoke the image encoder for shared image regions and tie the crops to specific cell layouts. We therefore partition images into tiles that are defined independently of cells, encode each tile once, and allow cells to retrieve their required tiles. This section describes the tiles, the feature bank that stores their features at inference, and the use of the tiles in training.

Tile pyramid. For each source image i, level ℓ of the pyramid is the image resized by $1/2^\ell$ from the original with Lanczos filtering. The deepest level L_max is the last one whose shorter side is at least 64 pixels. We cover each level with 512×512 tiles at a stride of 512 pixels, shifting the last tile along each axis inward to keep it inside the image. An axis shorter than 512 pixels gets one centered tile padded with black. A tile

$$\tau = (i, \ell, a, b) \tag{20}$$

is identified by its source image, level, and crop origin (a, b), and does not depend on any cell. It is treated as a virtual camera with intrinsics

$$K_\tau = \begin{pmatrix} f_{x,i}/2^\ell & 0 & c_{x,i}/2^\ell - a \\ 0 & f_{y,i}/2^\ell & c_{y,i}/2^\ell - b \\ 0 & 0 & 1 \end{pmatrix} \tag{21}$$

and the pose of its source image. For each cell, only this pose is transformed into the cell coordinates. The attention of Appendix D.1 treats every selected tile as one view, so a photograph can contribute several tiles to the same cell, at different crop origins or levels. At inference, a cell selects its tiles as described in Appendix D.3.

Feature bank at inference. Before sampling a cascade level, we identify the set of tiles required by its cells. Each photograph is cropped at the required pyramid levels and processed by the frozen DINOv3 and ConvNeXt encoders. The bank stores, in BF16,

$$\mathcal{B}[\tau] = \left(F_\tau^{\mathrm{DINO}}, F_{\tau,4}^{\mathrm{ConvNeXt}}, F_{\tau,8}^{\mathrm{ConvNeXt}}, F_{\tau,16}^{\mathrm{ConvNeXt}}\right), \tag{22}$$

where F_τ^{DINO} is the DINO token sequence and the other entries are the ConvNeXt feature maps at strides 4, 8, and 16. The SS and mesh flows use the same DINOv3 checkpoint, so DINO runs once per tile for both. The condition of a cell references the stored features $\mathcal{B}[\tau]$ of its selected tiles instead of copying them, so overlapping cells and all flow steps of both stages share a single encoding per tile. The bank is built per cascade level, as finer levels instantiate different cells and require different tiles.

Tiles in training. Because tiles are defined independently of cells, we share a single set of tiles across all cell layouts in training. Cells of every physical size and position, as well as the cell-frame augmentations of Appendix E.5, read the same tiles, so the stored tiles remain unchanged as the number of 3D training variants grows. For each cell, we take the tiles at the level closest to 8 pixels per SS voxel and draw 16 of them at random for each example.

D.3 DA3-BASED LAYOUT AND VIEW SELECTION

Running the generator on a new capture requires two decisions outside the model: where to place the coarsest cells, and which tiles each cell reads. We make both decisions using depth from Depth Anything 3 (DA3) (Lin et al., 2026), run with input poses and intrinsics as in the DA3 baseline (Appendix B.4). We use DA3 solely for these decisions: we do not feed its depth or features to the flow models, fuse them into the output mesh, or use them to optimize it.

Cell placement. At the coarsest level, we discard the 10% of DA3 depths with lowest confidence, unproject the remainder, and place cells over the volume that these points occupy. For some scenes, we additionally use the camera hull or ground-truth footprint as in Appendix C.5. The volume is covered by cells on a half-cell-stride lattice. At finer levels, candidate cells are derived from the SS occupancy predicted at the parent level in world coordinates. We dilate this occupancy by one parent SS voxel before intersecting it with the child lattice, so that a small miss at the coarse level does not remove the region from all finer levels.

Visibility gate. We determine which views observe each voxel center p of the 16^3 SS latent grid of a cell. A view is considered to observe p if p projects within the image frame and is not occluded in the DA3 depth map. The occlusion test tolerates larger depth discrepancies where DA3 confidence is low. On city blocks, we ignore sky, people, and vehicles in this test, using masks predicted by an off-the-shelf segmentation model. The generator still receives the unmasked images.

Pyramid level. Let ν_{pi} be the width of one SS voxel in pixels of the full-resolution image,

$$\nu_{pi} = \frac{f_{x,i}\,(s/64)}{z_i(p)}. \tag{23}$$

We target $\nu_0 = 8$ pixels per voxel, so that a 512×512 tile spans the 64^3 SS grid, and use the same target when selecting tiles in training (Appendix D.2). The preferred and available levels are

$$\ell_{pi}^* = \log_2\left(\frac{\nu_{pi}}{\nu_0}\right), \qquad \ell_{pi} = \mathrm{clip}\big(\mathrm{round}(\ell_{pi}^*), 0, L_{\max}\big)\,. \tag{24}$$

Inside the pyramid, level ℓ_{pi} shows the voxel at approximately 8 pixels. The residual $|\ell^*_{pi} - \ell_{pi}|$ measures how far the available level deviates from this scale.

Tile ranking and selection. We select tiles by how well they observe the cell. For every voxel center p and every view i that observes it, we compute the visibility score

$$S(p, i) = |\cos\theta_{pi}| - |\ell^*_{pi} - \ell_{pi}|, \tag{25}$$

where θ_{pi} is the angle between the viewing ray and the surface normal estimated from the DA3 depth. The first term favors views facing the surface, and the second penalizes pyramid levels that deviate from the preferred scale. For each voxel center, we take its 16 highest-scoring views and add each score to the tile containing the projection of p in that view. The score of a tile is the sum of these contributions over the cell.

Each cell retains its highest-scoring tiles, up to $\max(64, N)$, where N is the number of input views. Because the network reads at most 64 tiles per cell, we sample 64 of them at random when more are kept and use this fixed set of tiles for all flow steps. A cell unobserved by any view receives an empty condition, identical to the unconditional pass in classifier-free guidance.

E Training Details

We train two models sharing the same architecture: a general-scene model evaluated on ScanNet++, Tanks and Temples, and real-world captures, and a city-scale model specialized for our city-block dataset. Both models start from the same initialization (Appendix E.1) and shared multi-view pretraining on TexVerse (Appendix E.2), differing only in their final scene-level fine-tuning data (Appendices E.3 and E.4). Both employ the same inference procedure (Appendix C). The remaining subsections detail scale and rotation augmentations (Appendix E.5) and optimization settings (Appendix E.6).

E.1 Model initialization

Both flows are initialized from the official TRELLIS.2 weights. The SS flow retains the original 30-block architecture. For the mesh flow, we prune the 30-block backbone down to 15 transformer blocks by removing the blocks that affect its output the least, reducing the denoiser from approximately 1.3B to 655M parameters while preserving the hidden dimension, attention heads, and latent resolution of the retained blocks. To recover from pruning, we fine-tune the pruned mesh flow with the official single-image conditioning for 50k steps on 16 H200 GPUs with a learning rate of 3×10^{-5}. The voxel-queried image attention and the ConvNeXt branch are subsequently trained during image-conditioned multi-view training.

We freeze the pretrained SS and shape VAEs, which encode the clean SS and shape targets and decode generated latents. We update only the two flow networks and their conditioning layers during training.

E.2 Shared multi-view pretraining

After initialization, we train both flows on multi-view renderings of TexVerse (Zhang et al., 2025). This stage adapts the single-object generators, alongside the new conditioning modules, to posed multi-view inputs. The resulting checkpoint serves as the starting point for scene-level fine-tuning (Appendices E.3 and E.4). Training follows a curriculum over mesh resolution and the number of randomly sampled conditioning views. The mesh flow is trained first with 4 views at mesh resolution 256^3, then with 4 views at 512^3, and finally with 16 views at 512^3. The SS flow, whose occupancy targets have a resolution of 64^3, is trained with 4 and subsequently 16 views. Conditioning images are rendered or cropped to 512×512.

E.3 General-scene fine-tuning

Starting from the TexVerse checkpoint (Appendix E.2), we fine-tune both flows of the general-scene model on multi-scale scene data. In addition to the single-room scene datasets SAGE-10K (Xia



Figure 19: Examples of our procedural scene generation. The first two images show a synthetic multi-room house, and the last two show procedural grass scenes.

et al., 2026) and 3D-FRONT (Fu et al., 2021), we construct two procedural datasets: connected multi-room houses and outdoor ground with detailed geometry such as grass.

Synthetic multi-room houses. Fine-tuning solely on single-room datasets caused missing walls and ceilings on multi-room ScanNet++ scenes. We therefore generate 200 synthetic houses (Figure 19). For each house, we generate a floor plan with three to six connected rooms, some of them non-rectangular, and build walls, floors, ceilings, doors, windows, and furnishings from it. We allocate cameras to rooms in proportion to floor area and sample their height, pitch, and roll from the empirical distributions of ScanNet++ DSLR captures.

Procedural grass scenes. Existing room datasets contain little outdoor ground cover, causing models trained without it to over-smooth grass geometry. We generate 100 scenes of 32 m extent with procedural vegetation over displaced terrain (Figure 19). We model the vegetation as mesh geometry rather than image textures and vary its height, density, tilt, and coloration across the scene with spatial noise. We also place TexVerse objects at random positions with varied scales and orientations in each scene, and capture the scene with cameras on spiral trajectories.

Rendering. Every scene is rendered from 256 views. SAGE-10K, 3D-FRONT, and synthetic houses use 1752×1168 images with horizontal fields of view from $108.7°$ to $124.3°$, matching ScanNet++ DSLR parameters. Grass scenes use 3840×2560 images with a $56.0°$ horizontal field of view, matching Mip-NeRF 360 captures. Renderings are stored as 512×512 tiles for conditioning (Appendix D.2).

Data mixture. The sampling shares among training sources are:

Training source	Scenes	Sampling share
SAGE-10K	2,000	0.40
3D-FRONT	997	0.20
Synthetic houses	200	0.25
Procedural grass	100	0.15

Within each source, the sampling share is divided equally among its cell sizes. Room and house scenes use the six cell sizes $\{1, 1.5, 2, 3, 4, 6\}$ m, and grass scenes use $\{1, 2, 4\}$ m.

E.4 City-scale fine-tuning

The city-scale model is fine-tuned from the TexVerse checkpoint (Appendix E.2) on our city-block dataset (Appendix B.1), which pairs ground-level photographs of urban blocks with mesh targets derived from BIMs. It uses the same representations, voxel-queried image conditioning, multi-scale cell training, and inference procedure as the general-scene model, differing only in the fine-tuning data to accommodate urban blocks and building exteriors.

E.5 Scale and rotation augmentation of cells

We augment training cells in both scale and orientation.

Scale. We assign each room or house scene one of the two scale ladders $\{1, 2, 4\}$ m and $\{1.5, 3, 6\}$ m and partition it into cells of those sizes, which gives six cell sizes across the dataset. Because a scene yields far more small cells than large ones, we sample cells with the scale-balanced scheme of Appendix C.2 to avoid over-representing small cells.

Rotation. We rotate each cell independently of its scale in two ways. First, we rotate the cell about the vertical axis by a uniformly sampled angle and tilt the vertical axis by up to 10°. We sample the tilt angle from a distribution concentrated near zero, which keeps most cells nearly upright while making the model robust to camera poses that are not perfectly level. Second, we apply rotations by multiples of 90° about the vertical axis alongside reflection flips, yielding eight discrete orientations per cell. The same transform is applied to the target geometry and the camera poses, and reflections also flip the images. Because these eight discrete transforms only permute the voxel grid, we apply them directly to the dual-grid representation of TRELLIS.2 (Xiang et al., 2026) instead of recomputing it from the mesh. We still re-encode the SS and shape latents of each transformed cell, because neither VAE encoder is equivariant to these transforms. Because the tile bank is independent of cells (Appendix D.2), these augmentations leave the image processing unchanged: all cell variants access the same tiles and differ only in their relative camera transforms.

E.6 Optimization settings

Both flows are optimized using AdamW with a constant learning rate, weight decay of 0.01, $\beta = (0.9, 0.95)$, and $\epsilon = 10^{-8}$, with BF16 mixed precision and a per-GPU batch size of 8. The learning rate is 10^{-4} during the 4-view phases of the mesh flow, 5×10^{-5} in other pretraining phases, and 10^{-5} during scene-level fine-tuning. For classifier-free guidance, the conditioning input is dropped with probability 0.1. We set $\sigma_{\min} = 10^{-5}$ in the flow path of Equation 2. TexVerse pretraining (Appendix E.2) runs on 16 to 32 NVIDIA H200 GPUs for approximately 80k steps for the SS flow and 200k steps for the mesh flow in total. Scene-level fine-tuning (Appendices E.3 and E.4) runs for 25,000 steps per flow on 8 H200 GPUs.

F Scalable Mesh Decoding

F.1 Margin-based chunk decoding and stitching

We decode each cascade level into a unified scene mesh. Decoding cells independently would truncate the shape VAE receptive field at cell boundaries, producing visible seams. Instead, we assemble the shape tokens across all cells into a single sparse tensor for the level. Because cells overlap by 50% during MultiDiffusion (Appendix C.3), we select every other cell along each axis to partition the volume without overlap.

Because decoding the entire level at once exceeds GPU memory, we process it in blocks of 4^3 cells. Each block is padded with a 4-voxel latent margin from its neighbors, so that voxels near the block boundary are decoded with their surroundings. Each block outputs only the mesh faces belonging to its interior region, preventing duplicate faces between adjacent blocks. We merge the copies of each boundary vertex produced by adjacent blocks into a single vertex, identifying them by their integer voxel coordinates. The blocks thus join into one mesh without duplicate faces or open seams.

F.2 Combining cascade levels

In object-centered captures such as Mip-NeRF 360 (Barron et al., 2022), cameras orbit a central subject observed closely from many angles, while the background is seen only from afar. Because the background appears too small in the images to show fine detail, we do not reconstruct it at the finest resolution: we run the coarsest level over the full bounding volume and restrict successive refinement levels to concentric squares centered on the subject. For the `garden` scene in Figure 1, we employ three cascade levels, with each finer level covering half the horizontal extent of its predecessor.

Each level yields an independent mesh, which we merge hierarchically. Because finer meshes provide higher fidelity in the central region, we remove coarse faces whose centroids fall within the boundary of the next finer level, and combine the remaining outer ring with the finer mesh. This

ring composition is used only for the nonuniform qualitative reconstructions; on ScanNet++ and Tanks and Temples, all levels share identical spatial extents, and the finest level is evaluated directly.

G Additional Ablations

G.1 Parent initialization

Table 7 compares $t_s \in \{0.8, 0.9, 1\}$ on all 50 ScanNet++ scenes with 32 and 256 input views, extending Figure 5(a), which shows only the 25 large scenes with 32 views. We change t_s at every finer level. At $t_s = 1$, every finer level starts from pure noise, inherits only the cell placement, and runs the full schedule of 12 flow steps instead of $K = 10$. All other experiments use $t_s = 0.8$.

Parent initialization is most effective in wide scenes with sparse views. On the large scenes with 32 views, $t_s = 0.8$ improves every metric over $t_s = 1$: it raises F@2 from 0.6306 to 0.7155 and reduces the Chamfer distance by $5.0\times$.

In the other three settings, where the views cover the scene more densely, $t_s = 1$ reaches a higher F@2. Without the parent, the finest level does not inherit the errors of the coarser level, so it can place the surface more accurately where the views constrain it. The parent, however, still makes the reconstruction more consistent as a whole. With $t_s = 0.8$, the Chamfer distance drops by $2.9\times$ on the small scenes with 32 views and by $1.6\times$ on the large scenes with 256 views. NC is higher with $t_s = 0.8$ in every setting.

Table 7: **Effect of parent initialization on ScanNet++.** Every finer cascade level starts from a latent noised to t_s, where $t_s = 1$ generates from pure noise. Mean over the 25 smallest and the 25 largest scenes, with the best results in bold.

	Small (25)						Large (25)					
	2D			3D virtual scan			2D			3D virtual scan		
t_s	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑
32 Views												
0.8	**0.0324**	**15.31**	**0.9990**	**0.0190**	0.6718	**0.9296**	**0.0565**	**16.33**	**0.9962**	**0.0212**	**0.7155**	**0.9247**
0.9	0.0715	16.97	0.9909	0.0394	0.6730	0.8945	0.1450	19.16	0.9773	0.0621	0.6779	0.8580
1.0	0.0998	18.31	0.9823	0.0548	**0.7255**	0.8646	0.2170	22.15	0.9535	0.1051	0.6306	0.7881
256 Views												
0.8	**0.0361**	**15.49**	**0.9991**	**0.0234**	0.6167	**0.9260**	**0.0557**	**15.62**	0.9902	**0.0202**	0.7176	**0.9283**
0.9	0.0396	15.60	0.9967	0.0240	0.6487	0.9241	0.0619	16.12	**0.9955**	0.0268	0.7515	0.9190
1.0	0.0371	15.53	0.9913	0.0235	**0.7849**	0.9201	0.0684	16.54	0.9898	0.0315	**0.7820**	0.9081

Tanks and Temples with 16 input views at 4K shows the same trade-off as the denser ScanNet++ settings (Table 8): on average, $t_s = 0.8$ gives a slightly lower F@τ than $t_s = 1$ (0.2756 vs. 0.2830) but a clearly lower CD/τ (9.69 vs. 12.01). Figure 20 shows that at $t_s = 1$ the reconstruction loses consistency: floaters appear and the ground has holes.

Table 8: **Effect of parent initialization on Tanks and Temples with 16 input views.** Every finer cascade level starts from a latent noised to t_s, where $t_s = 1$ generates from pure noise. Inputs are the 4K images. F@τ and CD/τ use the official per-scene threshold τ, and CD is in meters. The best results are in bold and the second best underlined.

	Barn		Caterpillar		Courthouse		Ignatius		Meetingroom		Truck		Mean	
t_s	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD/τ↓
0.8	0.2080	0.0895	0.1949	0.0637	0.1159	**0.5905**	**0.4704**	**0.0055**	**0.3084**	**0.0476**	0.3563	0.0311	0.2756	**9.6855**
0.9	**0.2336**	**0.0831**	0.1960	0.0575	**0.1169**	0.6717	0.4436	0.0058	0.3071	0.0884	0.3850	**0.0304**	0.2804	10.5876
1.0	0.2304	0.1030	**0.2057**	**0.0555**	0.1084	0.6974	0.4584	0.0084	0.2654	0.1313	**0.4300**	0.0343	**0.2830**	12.0138

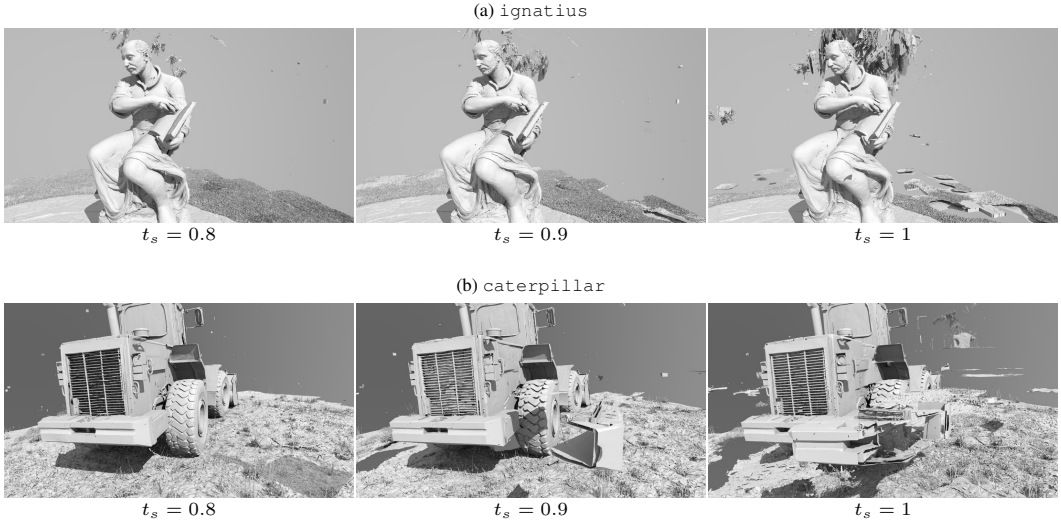


Figure 20: **Effect of parent initialization on Tanks and Temples.** Meshes at the finest cascade level from 16 input views at 4K, rendered from the same viewpoint per scene.

G.2 Image conditioning

Table 9 extends the comparison in Figure 6 to Tanks and Temples with 32 input views across all cascade levels. Voxel-queried attention slightly improves the mean F@τ over the baseline at every cascade level, while the ConvNeXt branch achieves the highest mean F@τ across all levels. Figure 21 compares the three conditioning variants on `ignatius` with 32 input views and on `caterpillar` with 256 input views. The baseline produces overly smooth geometry: it flattens the face and clothing folds of the statue, and merges the bars of the loader's rear grille into a continuous wavy surface. Voxel-queried attention better recovers facial features and clothing folds while separating several grille bars; the ConvNeXt branch sharpens these details further, cleanly resolving individual bars.

Table 9: **Effect of image conditioning on Tanks and Temples with 32 input views.** The baseline uses the GenRecon-style conditioner, our voxel-queried attention replaces it, and the ConvNeXt branch is then added. Inputs are the 4K images. F@τ and CD/τ use the official per-scene threshold τ, and CD is in meters. The best results are in bold and the second best underlined.

	Barn		Caterpillar		Courthouse		Ignatius		Meetingroom		Truck		Mean	
Conditioning	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD/τ↓
L0														
Baseline	**0.1781**	0.0691	0.1124	0.0839	0.0836	0.4428	0.2092	0.0113	0.3383	0.0360	0.2472	0.0344	0.1948	9.2747
Voxel attn.	0.1704	**0.0645**	0.1223	0.0814	0.0875	0.3684	0.2381	0.0100	0.3531	0.0359	0.2482	0.0347	0.2033	8.5584
+ ConvNeXt	0.1463	0.0787	**0.1328**	**0.0805**	**0.0965**	**0.3619**	**0.3223**	**0.0073**	**0.3561**	**0.0349**	**0.2955**	**0.0317**	**0.2249**	**8.4522**
L1														
Baseline	0.1734	0.0762	0.1439	0.0761	0.1146	0.3388	0.2918	0.0076	0.3579	0.0329	0.2659	0.0316	0.2246	8.0897
Voxel attn.	0.1819	0.0770	0.1524	0.0732	0.1038	0.3645	0.3175	0.0072	0.3647	0.0321	0.2668	0.0323	0.2312	8.1666
+ ConvNeXt	**0.1926**	**0.0641**	**0.1752**	**0.0727**	**0.1181**	**0.3355**	**0.4194**	**0.0058**	**0.3721**	**0.0318**	**0.3523**	**0.0305**	**0.2716**	**7.5968**
L2														
Baseline	0.1623	**0.0798**	0.1778	0.0720	0.1179	0.2642	0.3784	0.0063	0.3617	0.0320	0.2909	0.0314	0.2482	7.4231
Voxel attn.	**0.1944**	0.0887	0.1762	**0.0695**	0.1179	0.2607	0.3904	0.0061	0.3636	0.0319	0.2733	0.0311	0.2527	7.4387
+ ConvNeXt	0.1938	0.0866	**0.1945**	0.0740	**0.1324**	**0.2553**	**0.4904**	**0.0051**	**0.3806**	**0.0308**	**0.3705**	**0.0304**	**0.2937**	**7.4218**

(a) `ignatius`, 32 input views



Image | Baseline | Voxel attn. | + ConvNeXt

(b) `caterpillar`, 256 input views



Image | Baseline | Voxel attn. | + ConvNeXt

Figure 21: **Effect of image conditioning on Tanks and Temples.** Meshes at the finest cascade level from the 4K images, all rendered from the same viewpoint per scene. The baseline uses the GenRecon-style conditioner, our voxel-queried attention replaces it, and the ConvNeXt branch is then added.

G.3 TILE PYRAMID LEVEL SELECTION

For each view, a cell attends to the pyramid level matching its voxel spacing (Section 3.4). As detailed in Appendix D.3, we choose this level so that one SS voxel covers approximately 8 pixels (Equation 24). We compare this adaptive selection with two fixed levels shared across all cells: *Coarse* selects the coarsest level whose image height is at least 512 pixels (584 pixels on ScanNet++ and approximately 546 pixels on Tanks and Temples), whereas *Fine* uses the full-resolution images. Each configuration uses a separate model trained with the corresponding selection rule under the same schedule. We report every cascade level with 32 input views, on ScanNet++ in Table 10 and on Tanks and Temples in Table 11.

On ScanNet++ (Table 10), the images are 1168 pixels high, so *Coarse* halves them only once, to 584 pixels, and keeps most of their resolution. *Coarse* is therefore competitive on this dataset: it achieves a slightly higher F@2 and a similar CD at levels 1 and 2. The adaptive selection still yields the lowest MAE across all cascade levels and scene groups, and *Fine* underperforms the adaptive selection in nearly every column.

In contrast, on Tanks and Temples (Table 11), the inputs are 4K images about 2180 pixels high, so *Coarse* downsamples them by a factor of four, discarding high-frequency details needed by finer cells. As a result, *Coarse* falls behind at level 2, with a mean F@τ of 0.2305 compared to 0.2937. While *Fine* performs best on certain scenes (e.g., Barn), the adaptive selection achieves the best mean F@τ and CD$/\tau$ across all cascade levels.

Table 10: **Effect of tile pyramid level selection on ScanNet++ with 32 input views.** Coarse fixes every cell to the coarsest pyramid level whose image height is at least 512 pixels, and Fine to the full-resolution level; ours matches the level to the cell's voxel spacing. Mean over the 25 smallest, the 25 largest, and all 50 scenes, with the best results in bold.

	Small (25)						Large (25)						All (50)					
	2D			3D virtual scan			2D			3D virtual scan			2D			3D virtual scan		
Tile level	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑
L0																		
Coarse	0.0456	18.85	0.9984	0.0241	0.5633	0.9073	0.0718	**19.14**	**0.9986**	**0.0231**	**0.6349**	**0.9097**	0.0587	19.00	**0.9985**	0.0236	0.5991	0.9085
Adaptive (ours)	**0.0438**	**18.74**	**0.9986**	**0.0230**	**0.5816**	**0.9110**	**0.0714**	19.21	0.9984	0.0235	0.6336	0.9090	**0.0576**	**18.98**	**0.9985**	**0.0233**	**0.6076**	**0.9100**
Fine	0.0578	20.01	0.9967	0.0313	0.4919	0.8894	0.0892	19.89	0.9977	0.0275	0.5849	0.8966	0.0735	19.95	0.9972	0.0294	0.5384	0.8930
L1																		
Coarse	0.0355	16.68	0.9988	**0.0199**	**0.6476**	0.9227	0.0631	17.26	**0.9991**	**0.0198**	**0.7049**	**0.9232**	0.0493	16.97	**0.9989**	**0.0199**	**0.6762**	0.9229
Adaptive (ours)	**0.0346**	**16.35**	**0.9994**	0.0200	0.6359	**0.9240**	**0.0569**	17.10	0.9983	0.0199	0.6959	**0.9232**	**0.0457**	**16.72**	0.9988	**0.0199**	0.6659	**0.9236**
Fine	0.0407	17.23	0.9980	0.0239	0.5893	0.9144	0.0647	**17.03**	0.9981	0.0219	0.6747	0.9185	0.0527	17.13	0.9981	0.0229	0.6320	0.9164
L2																		
Coarse	0.0335	15.71	0.9980	**0.0190**	**0.6736**	0.9270	0.0639	16.97	**0.9972**	**0.0200**	**0.7163**	0.9237	0.0487	16.34	**0.9976**	**0.0195**	**0.6950**	0.9254
Adaptive (ours)	**0.0324**	**15.31**	**0.9990**	**0.0190**	0.6718	**0.9296**	**0.0565**	16.33	0.9962	0.0212	0.7155	**0.9247**	**0.0445**	**15.82**	**0.9976**	0.0201	0.6936	**0.9272**
Fine	0.0353	15.81	0.9923	0.0219	0.6265	0.9220	0.0605	**16.20**	0.9942	0.0209	0.7031	0.9232	0.0479	16.00	0.9932	0.0214	0.6648	0.9226

Table 11: **Effect of tile pyramid level selection on Tanks and Temples with 32 input views.** Coarse fixes every cell to the coarsest pyramid level whose image height is at least 512 pixels, and Fine to the full-resolution level; ours matches the level to the cell's voxel spacing. Inputs are the 4K images. F@τ and CD/τ use the official per-scene threshold τ, and CD is in meters. The best results are in bold.

	Barn		Caterpillar		Courthouse		Ignatius		Meetingroom		Truck		Mean	
Tile level	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD↓	F@τ↑	CD/τ↓
L0														
Coarse	0.1554	0.0740	0.1273	0.0730	0.0908	1.0964	0.2996	0.0086	0.3052	0.0365	**0.3289**	**0.0296**	0.2178	13.0521
Adaptive (ours)	0.1463	0.0787	**0.1328**	0.0805	**0.0965**	**0.3619**	**0.3223**	**0.0073**	**0.3561**	**0.0349**	0.2955	0.0317	**0.2249**	**8.4522**
Fine	**0.1675**	**0.0619**	0.1176	**0.0648**	0.0921	0.7501	0.2672	0.0091	0.3274	0.0390	0.2671	0.0348	0.2065	10.5084
L1														
Coarse	0.1831	0.0849	0.1498	0.0703	0.1021	0.7583	0.3243	0.0077	0.3199	0.0362	**0.3550**	**0.0286**	0.2390	10.8031
Adaptive (ours)	0.1926	0.0641	**0.1752**	0.0727	**0.1181**	**0.3355**	**0.4194**	**0.0058**	**0.3721**	**0.0318**	0.3523	0.0305	**0.2716**	**7.5968**
Fine	**0.2170**	**0.0579**	0.1488	**0.0641**	0.1164	0.6199	0.4006	0.0061	0.3518	0.0350	0.3367	0.0313	0.2619	9.1992
L2														
Coarse	0.1700	0.0800	0.1442	0.0682	0.1020	0.3401	0.3063	0.0081	0.3026	0.0396	0.3578	**0.0287**	0.2305	7.9414
Adaptive (ours)	0.1938	0.0866	**0.1945**	0.0740	**0.1324**	**0.2553**	**0.4904**	**0.0051**	**0.3806**	**0.0308**	**0.3705**	0.0304	**0.2937**	**7.4218**
Fine	**0.2002**	**0.0604**	0.1814	**0.0607**	0.1269	0.4057	0.4862	**0.0051**	0.3662	0.0343	0.3593	0.0304	0.2867	7.6054

G.4 Attention window size

We compare two window sizes for the voxel-queried image attention described in Appendix D.1: $w = 6$ and $w = 2$. Each window size is evaluated with a separate model trained under the same schedule. Table 12 shows that both settings perform comparably across all cascade levels with 32 and 256 input views. We therefore adopt $w = 2$, which attends to 4 tokens per view instead of 36, speeding up sampling in the mesh stage by roughly $4\times$.

Table 12: **Effect of the attention window size w on ScanNet++.** Each voxel attends to the $w \times w$ DINO tokens around its projection in every view. Mean over the 25 smallest, the 25 largest, and all 50 scenes, with the best results in bold.

	Small (25)						Large (25)						All (50)					
	2D			3D virtual scan			2D			3D virtual scan			2D			3D virtual scan		
Window	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑	MAE↓	Ang↓	Comp↑	CD↓	F@2↑	NC↑
32 Views, L0																		
$w = 6$	**0.0437**	**18.65**	0.9985	**0.0230**	**0.5845**	**0.9111**	0.0723	19.27	**0.9984**	0.0238	0.6316	**0.9090**	0.0580	**18.96**	0.9984	0.0234	**0.6080**	**0.9100**
$w = 2$ **(ours)**	0.0438	18.74	**0.9986**	**0.0230**	0.5816	0.9110	**0.0714**	**19.21**	**0.9984**	**0.0235**	**0.6336**	**0.9090**	**0.0576**	18.98	**0.9985**	**0.0233**	0.6076	**0.9100**
32 Views, L1																		
$w = 6$	**0.0345**	16.38	**0.9994**	**0.0200**	**0.6392**	**0.9245**	0.0580	17.22	**0.9985**	0.0201	**0.6968**	0.9230	0.0463	16.80	**0.9990**	0.0200	**0.6680**	**0.9237**
$w = 2$ **(ours)**	0.0346	**16.35**	**0.9994**	**0.0200**	0.6359	0.9240	**0.0569**	**17.10**	0.9983	**0.0199**	0.6959	**0.9232**	**0.0457**	**16.72**	0.9988	**0.0199**	0.6659	0.9236
32 Views, L2																		
$w = 6$	**0.0323**	15.51	0.9988	**0.0189**	0.6712	0.9293	0.0592	16.47	**0.9966**	0.0229	0.7129	0.9213	0.0458	15.99	**0.9977**	0.0209	0.6921	0.9253
$w = 2$ **(ours)**	0.0324	**15.31**	**0.9990**	0.0190	**0.6718**	**0.9296**	**0.0565**	**16.33**	0.9962	**0.0212**	**0.7155**	**0.9247**	**0.0445**	**15.82**	0.9976	**0.0201**	**0.6936**	**0.9272**
256 Views, L0																		
$w = 6$	0.0473	19.11	0.9987	0.0282	**0.4909**	0.9078	0.0689	19.05	0.9989	0.0234	0.6229	0.9122	0.0581	19.08	0.9988	0.0258	0.5569	0.9100
$w = 2$ **(ours)**	**0.0469**	**18.98**	**0.9988**	**0.0279**	0.4897	**0.9086**	**0.0683**	**18.95**	**0.9990**	**0.0233**	**0.6251**	**0.9125**	**0.0576**	**18.97**	**0.9989**	**0.0256**	**0.5574**	**0.9106**
256 Views, L1																		
$w = 6$	0.0379	**16.43**	0.9993	0.0242	0.5648	0.9210	**0.0570**	**16.58**	**0.9896**	**0.0216**	**0.6902**	**0.9229**	0.0474	**16.51**	**0.9945**	0.0229	0.6275	**0.9219**
$w = 2$ **(ours)**	**0.0375**	16.44	**0.9995**	**0.0238**	**0.5726**	**0.9211**	0.0571	16.68	0.9892	0.0217	0.6874	0.9224	**0.0473**	16.56	0.9943	**0.0227**	**0.6300**	0.9217
256 Views, L2																		
$w = 6$	0.0364	15.55	**0.9993**	0.0239	0.6037	0.9251	**0.0546**	15.69	**0.9912**	**0.0202**	0.7172	0.9282	**0.0455**	15.62	**0.9952**	0.0221	0.6604	0.9267
$w = 2$ **(ours)**	**0.0361**	**15.49**	0.9991	**0.0234**	**0.6167**	**0.9260**	0.0557	**15.62**	0.9902	**0.0202**	**0.7176**	**0.9283**	0.0459	**15.56**	0.9947	**0.0218**	**0.6672**	**0.9271**

G.5 Synthetic training data

We compare the model trained on our full data mixture against a variant trained without synthetic multi-room houses and grass scenes—i.e., using only SAGE-10K and 3D-FRONT (Appendix E.3). This comparison uses $w = 6$ instead of the final $w = 2$; the two window sizes perform on par (Appendix G.4). Figure 22 compares the finest-level reconstructions using 32 input views. Without synthetic multi-room houses, walls can go missing in multi-room scenes. In the ScanNet++ scene shown at the top of Figure 22, the model trained without them fails to reconstruct the left and far walls, whereas the full model reconstructs them. On `caterpillar`, the model without grass scenes reduces the grassy ground to a flat, featureless surface, whereas the full model captures individual grass tufts.

(a) ScanNet++ `09c1414f1b`



GT | Full | w/o house & grass

(b) `caterpillar`

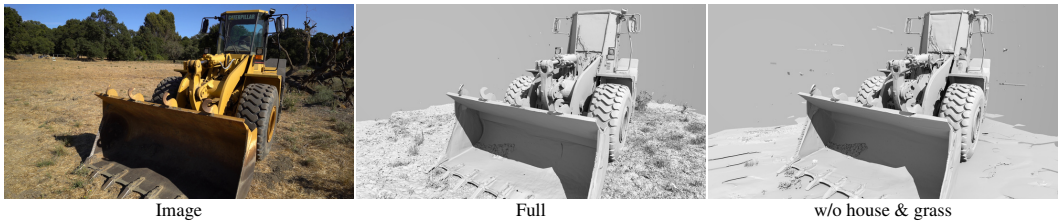


Image | Full | w/o house & grass

Figure 22: **Effect of the synthetic training data.** Meshes at the finest cascade level with 32 input views, from the model trained with our synthetic multi-room houses and grass scenes (Full) and from one trained on SAGE-10K and 3D-FRONT alone (w/o house & grass). Without the synthetic data, walls of the multi-room ScanNet++ scene disappear, and the grass around the loader becomes a flat surface.

H Inference Time

Table 13 reports the inference time for each cascade level on five large ScanNet++ scenes with 32 input views on a single NVIDIA H200 GPU. Because each level halves the cell size, the number of cells and the total computation can grow by roughly $8\times$ per level, and by more depending on the cell layout. In practice, the number of cells grows from 31 to 470 to 2,490, and runtime increases by $8.2\times$ from level 0 to level 1 but only by $4.7\times$ from level 1 to level 2, because finer levels allocate cells only where the parent level detects surfaces. Reconstruction up to level 1 takes an average of 17 minutes per scene, and up to level 2 takes 89 minutes, which is comparable to per-scene optimization methods. Flow sampling in the SS and mesh stages accounts for most of the runtime at all levels, whereas view selection, tile preparation, and parent transfer require less than 3 minutes combined. Peak GPU memory usage is 79 GiB at the finest level.

Table 13: **Inference time per cascade level on ScanNet++.** Wall-clock seconds on one NVIDIA H200 GPU with 32 input views, averaged over the 5 large scenes. View sel. + tiles includes DA3 depth, per-cell view selection, tile cropping and tile feature encoding (DINOv3 and ConvNeXt); parent transfer decodes the parent level to initialize the child SS. Model loading is excluded. Peak is the peak allocated GPU memory.

	Count		Time (s)						Memory (GiB)
Level	#Cells	#Tiles	View sel. + tiles	Parent transfer	SS	Shape	Decode	Total	Peak
L0 (4 m)	31	329	23	–	38	48	2.4	112	36
L1 (2 m)	470	473	26	21	466	394	7.6	914	57
L2 (1 m)	2,490	591	27	60	2,428	1,740	29	4,285	79
Total			76	82	2,932	2,181	39	5,310 (89 min)	79