SPHEREBRIDGE: SINGLE-PANORAMA 3D GAUSSIAN RECONSTRUCTION FOR HIGH-RESOLUTION REAL-TIME MULTI-SPHERE RENDERING

Anonymous authors
Paper under double-blind review

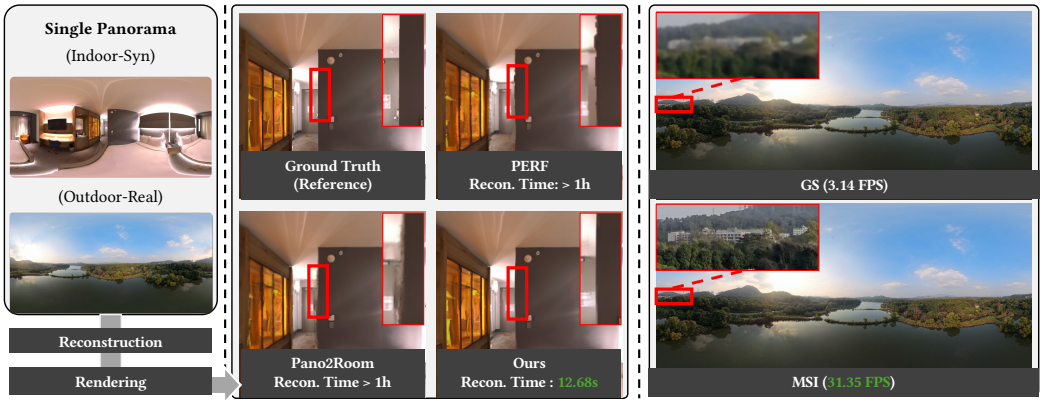

Figure 1: **SphereBridge** predicts a panorama-wide 3D Gaussian field from a single panorama in one forward pass and bakes it into a Multi-Sphere Image (MSI) for high-resolution nearby-view browsing. **(a)** Input panoramas and pipeline stages. **(b)** Reconstruction on a synthetic indoor scene against PERF and Pano2Room. **(c)** A 2048×4096 equirectangular nearby view of a real outdoor capture rendered by direct splatting (GS) and by the MSI.

ABSTRACT

Immersive 360-degree browsing in VR/AR and spatial photography requires turning a single panorama into a high-resolution scene representation that supports nearby views. Feed-forward 3D Gaussian Splatting (3DGS) has made nearby-view synthesis practical for perspective images, but its pinhole recipe assumes a planar local view, whereas a panorama covers the full sphere and must be reconstructed as one 360-degree scene. Tangent-crop workarounds restore pinhole compatibility at the cost of stitching seams and scale drift, and single-panorama pipelines that respect spherical geometry need over an hour of per-scene optimization. We present SphereBridge, a panorama-native pipeline that predicts a panorama-wide 3D Gaussian field from a single equirectangular panorama in one forward pass and bakes it into a Multi-Sphere Image (MSI) for high-resolution rendering. The Multi-Sphere Rendering Bridge routes the Gaussians onto concentric spherical layers, which reduces each frame to closed-form ray-sphere compositing with no per-frame Gaussian sorting, and injects the source panorama's high-frequency detail into the layer textures, which preserves detail at high output resolution. Because real captures rarely provide registered nearby views, training relies on synthetic data, and a training-free Tangent-View Scaffolding extends it to unlabeled real panoramas, using a stitched tangent-view reconstruction as a displaced input view and the real capture as supervision. Across indoor and outdoor benchmarks, SphereBridge reaches state-of-the-art single-panorama quality on LPIPS and DISTS, and its MSI renderer raises perceptual quality at high output resolution while rendering 4096×4096 perspective views at 23.92 FPS on a single NVIDIA A800 GPU.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Single-image novel-view synthesis recovers a 3D scene representation from one image and serves VR/AR and spatial photography. This paper focuses on the single-panorama setting: from one panorama, reconstruct a 3D representation that a viewer immersed in the captured scene can browse at high resolution across the nearby views produced by natural head motion. Compared with perspective single-image synthesis, this setting couples three requirements: reconstruction must cover the full sphere, rendering must sustain high resolution in real time from a representation that spans the whole sphere, and the model must generalize to in-the-wild panoramas, which come without registered nearby views.

Reconstruction faces a mismatch between perspective-native 3D Gaussian Splatting (3DGS) and panoramic imagery. Feed-forward methods such as SHARP (Mescheder et al., 2026) predict depth and Gaussians from one perspective image in one forward pass, but are built for planar projections, pinhole intrinsics, and bounded fields of view. A panorama instead covers the full sphere with spherical rays, wraps around horizontally, and folds across the poles, so a pinhole recipe fits neither its rays nor its borders. Tangent-crop workarounds restore pinhole compatibility but fragment the scene into local reconstructions with stitching seams, scale drift, and inconsistent geometry across crops. Panorama-specific pipelines such as PERF (Wang et al., 2024) and Pano2Room (Pu et al., 2024) respect spherical geometry but rely on per-scene optimization over diffusion-generated views and take over an hour per scene. Reconstruction therefore needs a panorama-native pipeline that handles spherical geometry in one forward pass.

Rendering the reconstructed Gaussians at high resolution is costly: direct 3DGS rasterization sorts and splats every Gaussian for every frame, so per-frame cost grows with both scene complexity and output resolution. In-the-wild generalization poses a different problem. Real panoramas rarely come with registered nearby views, so training relies on synthetic panoramas with rendered nearby views, and a training pair for a real capture has to be built from that capture alone.

We present SPHEREBRIDGE, a panorama-native pipeline that meets these requirements with three components (Figure 2). Panorama-Native Gaussian Reconstruction predicts a panorama-wide 3D Gaussian field from one equirectangular projection (ERP) input in one forward pass, back-projecting depth along spherical rays on an equal-area sphere and refining the Gaussians with a topology-aware decoder, so spherical rays and wrap-around borders need neither tangent crops nor per-scene optimization. The Multi-Sphere Rendering Bridge then bakes the Gaussians once into a Multi-Sphere Image (MSI), a set of textured concentric spherical layers, which reduces each frame to closed-form compositing of a fixed set of layers, and injects the source panorama's high-frequency appearance into the layer textures, which preserves detail at high output resolution. Tangent-View Scaffolding turns each unlabeled real panorama into a pseudo novel-view training pair by stitching tangent-view perspective reconstructions into a displaced input view, so seams and scale drift affect only the input while the real capture supervises the output.

Figure 1 previews the results: on a synthetic indoor scene, SphereBridge reconstructs in 12.68 s where PERF and Pano2Room take over an hour, and on a real outdoor capture, the MSI renders a 2048×4096 equirectangular nearby view at 31.35 FPS against 3.14 FPS for direct splatting, with sharper detail. Across indoor and outdoor benchmarks, SphereBridge reaches state-of-the-art single-panorama quality on LPIPS and DISTS, and the MSI renderer raises perceptual quality at high output resolution while rendering 4096×4096 perspective views at 23.92 FPS, all on a single NVIDIA A800 GPU. Our contributions are:

- **Panorama-Native Gaussian Reconstruction:** a feed-forward model that predicts a panorama-wide 3D Gaussian field from a single ERP input in one forward pass.
- **Multi-Sphere Rendering Bridge:** a one-time bake of the Gaussians into an MSI with panorama-derived layer textures for real-time high-resolution rendering.
- **Tangent-View Scaffolding:** a training-free construction that turns unlabeled real panoramas into pseudo novel-view training pairs for in-the-wild generalization.

2 Related Work

2.1 Perspective 3D Gaussian Reconstruction

3DGS (Kerbl et al., 2023) optimizes anisotropic Gaussians per scene from posed images, and feed-forward variants replace this optimization with a single network pass. PixelSplat (Charatan et al., 2024), MVSplat (Chen et al., 2024), and GS-LRM (Zhang et al., 2024) predict Gaussians from multiple posed views. Flash3D (Szymanowicz et al., 2025) and SHARP (Mescheder et al., 2026) predict them from one perspective image by lifting monocular depth into pixel-aligned Gaussians, and Multi-Layer Gaussian (Diao et al., 2025) and World Tracing (Zhang et al., 2026) model structure behind the visible surface with stacked Gaussian and point layers, respectively. All of these methods predict pixel-aligned primitives along pinhole rays on a planar image grid, so they assume a bounded field of view with no wrap-around border. A panorama violates both assumptions, and applying them to tangent crops fragments the scene into local reconstructions that must be stitched. SphereBridge keeps the feed-forward Gaussian recipe but back-projects depth along spherical rays and refines the Gaussians with a topology-aware decoder, so one forward pass covers the full sphere.

2.2 Panoramic 3D Reconstruction

Panoramic reconstruction methods respect spherical geometry but differ in the number of input panoramas and in how the scene is fitted. Single-panorama methods such as PERF (Wang et al., 2024) and Pano2Room (Pu et al., 2024) inpaint occluded regions with a Stable Diffusion prior (Rombach et al., 2022) and optimize a per-scene representation over the generated views, which takes over an hour per scene in multi-stage pipelines tuned to indoor scenes. Feed-forward methods such as Splatter-360 (Chen et al., 2025), PanSplat (Zhang et al., 2025), and PanoSplatt3R (Ren et al., 2025) predict Gaussians in one pass, but they take two or more wide-baseline panoramas and aggregate cross-view evidence to recover geometry, so a single panorama gives them no second view to match. SphereBridge combines the two settings. It predicts Gaussians from one panorama in a single forward pass, replacing cross-view matching with a panoramic depth prior.

2.3 Layered Representations for View Synthesis

Layered representations render a novel view by warping and alpha-compositing a fixed set of RGBA layers, so per-frame cost is independent of scene complexity. Multiplane Images (MPIs) place the layers on fronto-parallel planes: Stereo Magnification (Zhou et al., 2018) predicts them from a stereo pair, Single-View MPI (Tucker & Snavely, 2020) from one image, AdaMPI (Han et al., 2022) adapts the plane depths to in-the-wild scenes, and TMPI (Khan et al., 2023) tiles the layers to reduce redundancy. MSIs replace the planes with concentric spheres so the layers cover the full sphere, as in MatryODShka (Attal et al., 2020) for real-time 6DoF video and MSI-NeRF (Yan et al., 2025) for generalizable panoramic radiance fields. LayerPano3D (Yang et al., 2025) and Scene4U (Huang et al., 2025) decompose a single panorama into depth-clustered or semantic layers, inpaint occluded content with a generative model, and then optimize Gaussians per scene over the layers. SphereBridge reverses this order. It predicts the Gaussians first and uses the MSI only as a rendering bridge, baking the Gaussians onto spherical layers once, so rendering inherits the closed-form compositing of layered representations while reconstruction keeps the flexibility of 3DGS.

3 Method

Overview. As shown in Fig. 2, SphereBridge takes a single ERP panorama and produces a 360-degree scene for interactive high-resolution browsing. *Panorama-Native Gaussian Reconstruction* regresses panorama-wide Gaussians from the ERP input in one forward pass, and the *Multi-Sphere Rendering Bridge* bakes them into an MSI for high-resolution rendering. *Tangent-View Scaffolding* turns unlabeled real panoramas into training pairs for in-the-wild fine-tuning and also serves as a training-free perspective-stitching baseline.

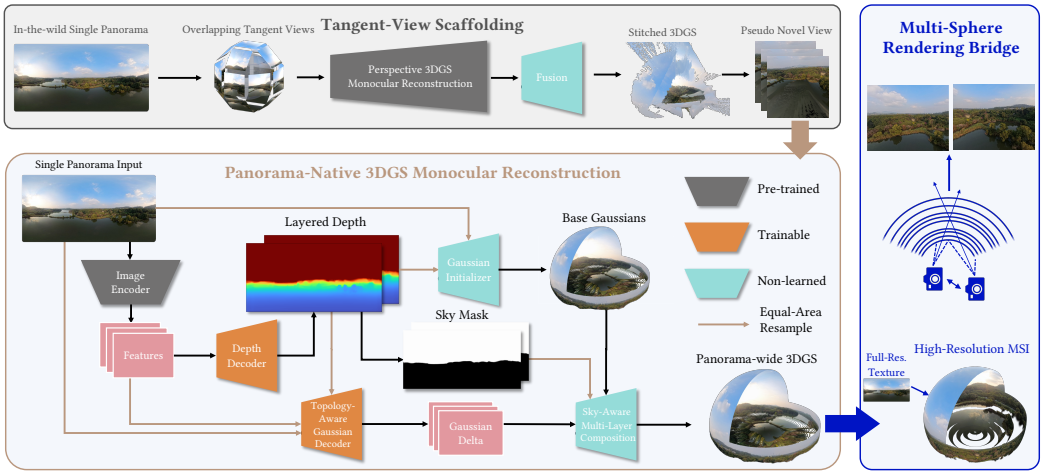

Figure 2: **SPHEREBRIDGE pipeline.** *Panorama-Native Gaussian Reconstruction* (bottom) maps a single ERP panorama to a panorama-wide 3DGS scene in one forward pass. Gray blocks are pre-trained and frozen, orange blocks are trained, and cyan blocks are non-learned operations. Brown arrows mark inputs resampled onto the equal-area lattice of Section 3.1. The *Multi-Sphere Rendering Bridge* (right) bakes the Gaussians once into an MSI whose layer textures take their detail from the source panorama, which enters the bridge directly, for high-resolution nearby-view rendering. *Tangent-View Scaffolding* (top) builds training pairs from unlabeled real panoramas. A stitched tangent-view reconstruction rendered at a pose displaced from the capture viewpoint forms the model input, and the captured panorama supplies the supervision.

3.1 PANORAMA-NATIVE GAUSSIAN RECONSTRUCTION

Panorama-Native Gaussian Reconstruction predicts a panorama-wide 3DGS scene from a single ERP input in one forward pass, following the bottom row of Fig. 2. The first step extracts DINOv3 features (Siméoni et al., 2026) from the ERP input with the frozen encoder of a panoramic depth model (Lin et al., 2026), and a trainable depth decoder predicts a two-layer metric depth map (Flynn et al., 2019) from these features. The features and the depth are then resampled onto a sphere-aware lattice. Every lattice sample is back-projected along its ray at both of its depths, initializing two base Gaussians per ray that share the sampled color and differ in depth and scale. A sky mask is also derived at this stage by thresholding the predicted depth. A topology-aware Gaussian decoder takes the encoder features together with the RGB, ray direction, depth, and sky mask of every sample, and predicts per-Gaussian deltas for color, opacity, inverse distance, angular offset, scale, and rotation. Applying these deltas to the base Gaussians in a sky-aware composer yields the final scene. The encoder stays frozen and only the two decoders are trained, with their module configuration given in Appendix C.1. The lattice and the composition are described next.

Sphere-Aware Lattice and Padding. The initializer and the decoder operate on a lattice that follows the sphere rather than the ERP grid. ERP pixels near the poles subtend far smaller solid angles than pixels near the equator, so one Gaussian per ERP pixel would crowd the poles and starve the equator. Equal-area resampling therefore maps every ERP-aligned input of the initializer and the decoder onto a fixed sampling pattern with uniform density on the sphere (Fig. 7), so Gaussian density is uniform in solid angle from the initialization onward. The ERP grid also has no image border: its left and right edges are one seam, and its top and bottom edges are the poles. Every convolution in the topology-aware decoder therefore uses

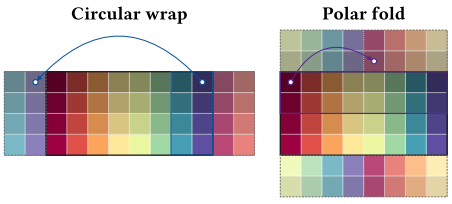

Figure 3: **Circular padding.** Circular wrap (left) crosses the left-right ERP seam, while polar fold (right) shifts longitude by π across the pole.

the circular padding of Jiang et al. (2025), a circular wrap along longitude and a polar fold across the poles (Fig. 3), so receptive fields cross both boundaries without seeing an artificial edge.

Sky-Aware Multi-Layer Composition. The composer applies the decoder deltas to the base Gaussians, with a separate branch for sky samples. A single Gaussian per ray can only cover the surface visible from the input viewpoint, whereas nearby views also expose surfaces occluded from it. The pre-split initialization therefore places two copies on every ray, and the inverse-distance and angular-offset deltas let each copy move along and off its ray. Where one copy already covers the visible surface, the other is surplus, and the network relocates such copies from well-covered surroundings into the occluded regions, which typically continue the geometry and texture of the visible surface around them. This pre-split with composition is a feed-forward analogue of 3DGS split densification: the initializer places redundant primitives and the composer specializes them. Sky samples take a separate branch because their depth is near infinite, where an unconstrained inverse-distance delta easily collapses training. The composer therefore holds their inverse distance at a far constant and applies all other deltas. The fixed distance costs nothing under nearby-view browsing, where the sky already behaves as an environment map at infinity, and the redundant sky copies can still fill sky regions occluded by foreground objects.

3.2 Multi-Sphere Rendering Bridge

The Multi-Sphere Rendering Bridge converts the reconstructed Gaussians into an MSI whose layer textures carry the full resolution of the source panorama. Direct 3DGS rasterization sorts and splats every primitive for every frame, and its detail is bounded by the colors of Gaussians reconstructed from a 512×1024 copy of the panorama. For nearby-view browsing, we therefore bake the Gaussians once into an MSI of 16 concentric spherical RGBA layers, spaced uniformly in inverse distance between the 1st and 99th percentiles of the reconstructed distance. The bake takes three steps, summarized in Algorithm 2 of Appendix E: routing assigns each Gaussian to layers and splits its optical depth, compositing rasterizes the routed Gaussians into layer textures at the resolution of the source panorama, and injection blends the panorama into each texture pixel (texel) according to its visibility from the source viewpoint.

Hybrid Soft-Hard Routing. The first step assigns each Gaussian t normalized layer weights $w_{p,t}$. Hard routing to the nearest layer in inverse distance keeps layer interiors crisp but leaves seams where a surface crosses a layer interface. Soft routing over neighboring layers removes the seams but copies foreground opacity onto a nearer layer with no content of its own, and the copy shifts under parallax into a ghost. Each Gaussian therefore takes the nearest layer as its home layer, and a disparity-space tent filter splits its weight between the home layer and the adjacent neighbor, with a boundary factor that is zero at the slab center and one at the interface, so softening stays local to layer boundaries. Two gates then screen the neighbor share, blocking copies of locally frontmost Gaussians on nearer layers and routes to a nearer layer with no local support, and blocked weight returns to the home layer, so the weights of each Gaussian still sum to one. Each weight finally scales the Gaussian's optical depth $\tau_t = -\log(1-\alpha_t)$ rather than its alpha, giving the layer copies $\alpha_{p,t} = 1-(1-\alpha_t)^{w_{p,t}}$, whose transmittances multiply back to $1-\alpha_t$ so that no coverage is lost, whereas a linear split of alpha would leave the Gaussian spuriously transparent.

Full-Resolution Layer Compositing. The second step rasterizes the routed Gaussians of each layer p from the sphere center and composites them front to back along the radial ray through every texel direction ω,

$$\alpha_p(\omega) = 1 - \prod_t \big(1-\alpha_{p,t}(\omega)\big), \qquad \bar{c}_p(\omega) = \frac{1}{\alpha_p(\omega)} \sum_t T_{p,t}(\omega)\, \alpha_{p,t}(\omega)\, c_t,$$

where $\alpha_{p,t}(\omega)$ is the routed opacity of Gaussian t under its footprint at ω and $T_{p,t}(\omega) = \prod_{s<t} \big(1-\alpha_{p,s}(\omega)\big)$ its in-layer transmittance. This layered composite approximately reproduces the source-view rendering of the Gaussian field, as derived in Appendix E. The textures are high-resolution but low-frequency, because a single Gaussian's footprint covers many texels.

High-Frequency Texture Injection. The third step is a single parameter-free pass over the source panorama I, whose pixels correspond one-to-one with the layer texels. Compositing the current MSI

from the sphere center yields the visibility $v_p(\omega) = \big(\prod_{q<p}(1-\alpha_q(\omega))\big)\,\alpha_p(\omega)$ of layer p along ω, and each texel color is blended toward the panorama in proportion to it,

$$c_p(\omega) \;\leftarrow\; v_p(\omega)\, I(\omega) + \big(1 - v_p(\omega)\big)\, \bar{c}_p(\omega),$$

while all alphas stay fixed. Visible surfaces, where $v_p \approx 1$, take their colors from the panorama at its native resolution. Occluded texels, where $v_p \approx 0$, keep the composited Gaussian colors, which come from laterally visible surroundings and expose correct but low-frequency content at disocclusions. Output detail is therefore bounded by the panorama rather than by the reconstruction, which only supplies geometry and visibility.

After the bake, the camera stays inside the shells for nearby views, so layer order follows sphere radius. Each pixel then reduces to 16 closed-form ray-sphere intersections composited front to back, with no per-frame sorting and a cost independent of the Gaussian count, as detailed in Appendix C.5 together with the routing parameters.

3.3 Tangent-View Scaffolding

Tangent-View Scaffolding turns an unlabeled real panorama into a training pair whose input and target come from different viewpoints, so that Panorama-Native Gaussian Reconstruction can be fine-tuned on in-the-wild captures without registered nearby views. The construction is training-free and also serves as the perspective-stitching baseline of Section 4.1. Given one ERP panorama, the scaffold estimates a panoramic depth map with the off-the-shelf panoramic depth model (Lin et al., 2026) and samples 18 overlapping tangent perspective views together with per-view projections of this depth map. Each view, paired with its projected depth, is reconstructed by a perspective single-view 3DGS regressor (Mescheder et al., 2026), so the projected depth anchors every per-view reconstruction in the shared metric frame. The per-view Gaussians are mapped into the world frame, and a scale alignment against the panoramic depth reduces the residual per-view scale error that survives independent reconstruction. Overlapping views are then blended at the Gaussian level by opacity weights that favor Gaussians near their view center, as detailed in Appendix C.2. The fused scene is rendered from a displaced pose into an ERP that serves as the model input, while the original capture at its own pose serves as the target. The input therefore carries the seams and scale drift of stitching, but the supervision is anchored to real pixels, so fine-tuning trains novel-view synthesis on real captures without injecting pseudo-label targets into the loss.

3.4 Two-Stage Training

Training proceeds in two stages, detailed in Appendix C. Stage 1 uses a multi-view panorama dataset, where each sample pairs a source panorama, the model input, with several nearby target panoramas. Stage 2 fine-tunes on the pseudo novel-view training pairs of Section 3.3, keeping the depth path frozen. In both stages, Panorama-Native Gaussian Reconstruction is trained through differentiable rendering against cubemap-face projections of the target panoramas. The objective combines three groups of terms. Rendered cubemap-face losses on RGB, depth, and alpha supervise appearance, geometry, and visibility. An ERP-weighted depth loss on the first depth layer anchors the trainable depth decoder, and SHARP-style regularizers (Mescheder et al., 2026) on large disparity gradients and the distant sky mask suppress floaters and refine the sky mask. Stage 2 drops the depth and sky-mask terms, which lack reliable supervision on real captures.

4 Experiments

4.1 Experimental Setup

Datasets and protocol. We evaluate on six benchmarks: Matterport3D (MP3D) (Chang et al., 2017) and Replica (Straub et al., 2019) for indoor scenes rendered from scanned meshes, 360Loc (Huang et al., 2024) for indoor real scenes, TartanGround (Patel et al., 2025) and PanoCity (Guo et al., 2026) for outdoor synthetic scenes, and OmniPhotos (Bertel et al., 2020) for outdoor real scenes. Every method receives the same source panorama, and target views are restricted to nearby views by a 20° parallax-angle filter computed from source-view depth, as detailed in Appendix A.

Table 1: Comparison with single-panorama methods on six benchmarks, with **best** and second-best highlighted and † marking indoor-only methods on outdoor cells.

Domain	Indoor						Outdoor						Recon.
Dataset	MP3D		Replica		360Loc		TartanGround		PanoCity		OmniPhotos		time
Method	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓	(s)↓
PERF	0.415	0.123	0.198	0.132	0.442	0.214	0.386†	0.204†	0.412†	0.223†	0.398†	0.156†	>1 h
Pano2Room	0.237	**0.088**	**0.192**	0.114	0.409	0.188	0.402†	0.200†	0.371†	0.190†	0.408†	0.153†	>1 h
Depth-Backproj	0.256	0.115	0.304	0.162	0.466	0.240	0.363	0.190	0.368	0.218	0.402	0.195	**0.26**
Tangent-View Scaffolding	**0.230**	0.096	0.240	0.144	0.429	0.225	0.312	0.167	0.374	0.221	0.359	0.169	63.92
SphereBridge	0.251	0.096	0.197	**0.085**	**0.351**	**0.111**	**0.283**	**0.123**	**0.237**	**0.107**	**0.317**	**0.094**	12.68

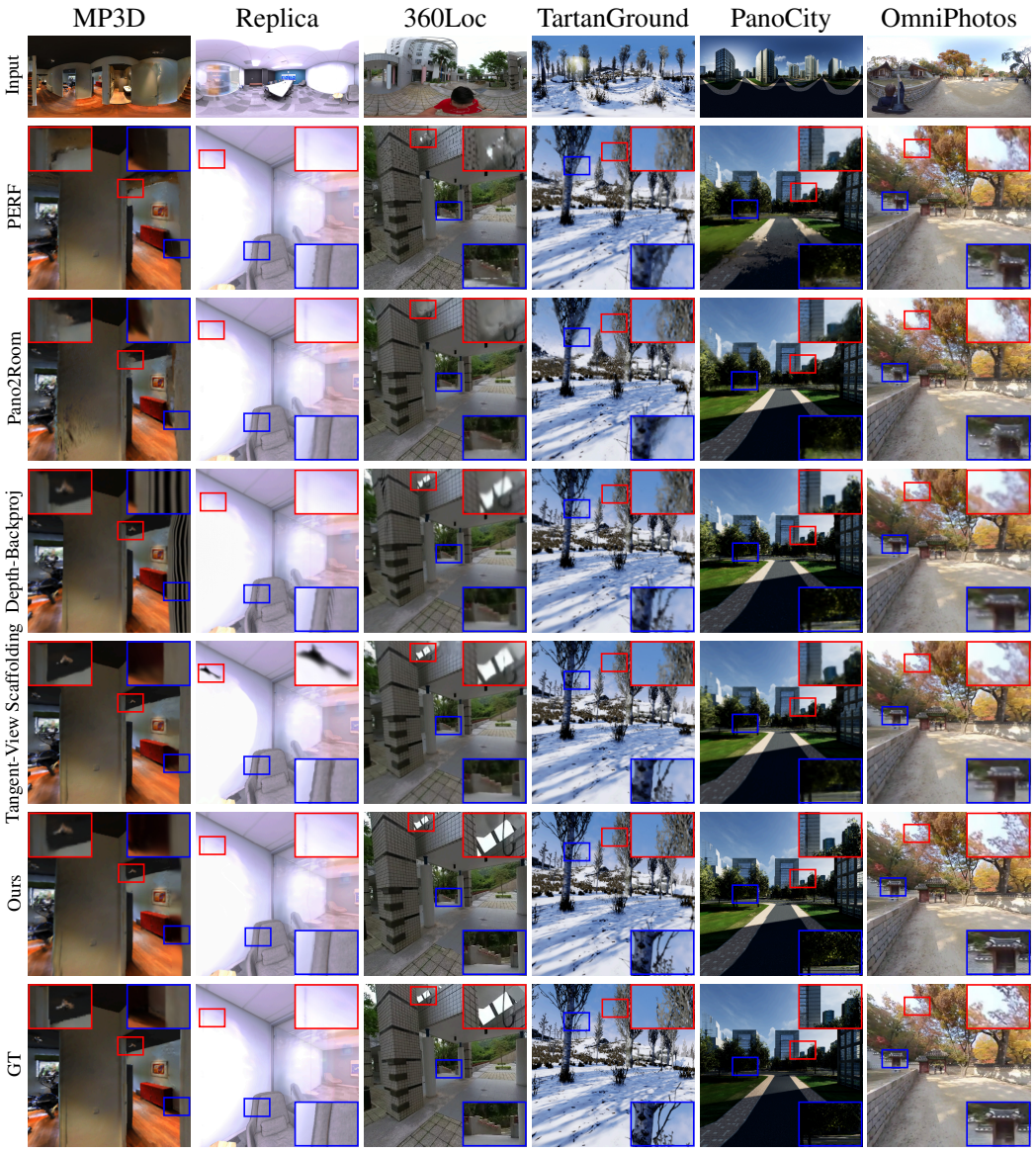

Figure 4: Qualitative comparison on single-panorama novel-view synthesis across six datasets, with the input panorama and the ground-truth novel view shown for reference.

Metrics. We report the perceptual metrics LPIPS (Zhang et al., 2018) and DISTS (Ding et al., 2022), following SHARP (Mescheder et al., 2026). Both rely on feature networks trained on perspective images, so we evaluate on cubemap faces rather than on ERP images, whose polar stretching would distort the features. Appendix B adds PSNR, SSIM, and ERP-side metrics per dataset.

Table 2: Rendering comparison on the same SphereBridge reconstruction of a texture-rich synthetic scene, with FPS on one NVIDIA A800 GPU.

Resolution	Output path	LPIPS↓	DISTS↓	FPS↑
640 × 640	Direct GS	0.400	0.211	111.29
640 × 640	Direct GS + Upsample	0.401	0.226	218.13
640 × 640	MSI Rendering	**0.365**	**0.152**	**1338.25**
1024 × 1024	Direct GS	0.431	0.232	53.97
1024 × 1024	Direct GS + Upsample	0.435	0.264	107.29
1024 × 1024	MSI Rendering	**0.372**	**0.148**	**701.69**
2048 × 2048	Direct GS	0.454	0.264	9.55
2048 × 2048	Direct GS + Upsample	0.454	0.320	30.37
2048 × 2048	MSI Rendering	**0.385**	**0.166**	**127.17**
4096 × 4096	Direct GS	0.479	0.334	1.77
4096 × 4096	Direct GS + Upsample	0.484	0.414	8.04
4096 × 4096	MSI Rendering	**0.423**	**0.235**	**23.92**

Table 3: Reconstruction-side ablations rendered directly from the predicted Gaussians, with the best per group in bold. The upper group removes one component of Section 3.1, and the lower group trains with and without Stage 2.

	TartanGround		OmniPhotos	
Variant	LPIPS↓	DISTS↓	LPIPS↓	DISTS↓
Full SphereBridge	**0.323**	**0.162**		
− Equal-area resampling	0.331	0.168		
− Circular padding	0.364	0.200		
− Sky-aware composition	0.351	0.188		
With Stage 2 (full)	0.323	0.162	**0.316**	**0.124**
Without Stage 2	**0.296**	**0.148**	0.337	0.131

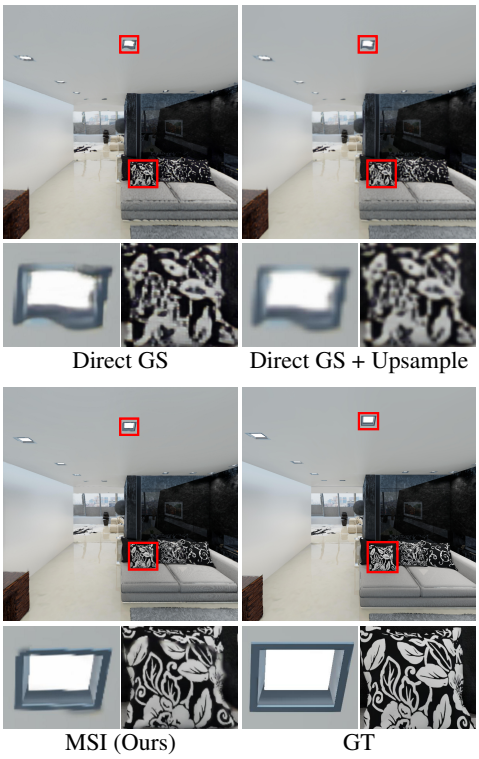

Figure 5: High-resolution rendering comparison at 4K on the same SphereBridge reconstruction.



Figure 6: Qualitative ablation of layer routing on the same reconstruction and target view.

Baselines. PERF (Wang et al., 2024) and Pano2Room (Pu et al., 2024) optimize a scene per panorama with diffusion-based inpainting and target indoor scenes. Depth-Backproj lifts a DAP (Lin et al., 2026) panoramic depth estimate into one Gaussian per pixel and serves as the depth-prior baseline. The Tangent-View Scaffolding of Section 3.3 serves as the perspective-stitching baseline, reconstructing each tangent view with SHARP (Mescheder et al., 2026) and fusing the per-view Gaussians under a shared panoramic depth prior.

Implementation. SphereBridge is trained on TartanGround with its evaluation scenes held out and fine-tuned on self-captured unlabeled panoramas with scaffolded inputs, as specified in Appendix C. At inference, the model reconstructs from a 512×1024 copy of the source panorama and bakes a 16-layer MSI at the source resolution.

4.2 Comparison with Single-Panorama Methods

Table 1 compares SphereBridge with the four baselines on perceptual quality and reconstruction time, with every cubemap face rendered at 640×640. SphereBridge reaches the lowest LPIPS and DISTS on all three outdoor datasets and wins the most entries on the indoor datasets. It reaches this quality in one feed-forward pass, which reduces reconstruction time from over an hour for PERF and Pano2Room to 12.68 s including the MSI bake.

Figure 4 shows where each baseline loses quality. PERF and Pano2Room inpaint unobserved regions with diffusion models, which hallucinate structures absent from the scene, and outside their indoor target setting, foliage degrades into feathery, dot-textured blobs. Depth-Backproj holds one Gaussian per source pixel and no occluded content, so boundary Gaussians stretch across disocclusions under viewpoint change and smear object edges. Tangent-View Scaffolding leaves stitching seams between tangent views, visible on the Replica wall. SphereBridge shows none of these artifacts and preserves fine texture and crisp boundaries on all six datasets. Appendix D extends the comparison to in-the-wild captures, and the supplementary material shows SphereBridge along nearby-view trajectories on these captures and on public panoramas from Poly Haven (2026).

4.3 Multi-Sphere Rendering vs. Direct Gaussian Splatting

On the same SphereBridge reconstruction, we compare MSI rendering with direct Gaussian splatting at the target resolution and a low-cost variant that splats at the 512×512 training resolution and upsamples in image space. Table 2 reports LPIPS, DISTS, and FPS at resolutions from 640×640 to 4096×4096. MSI rendering reaches the lowest LPIPS and DISTS and the highest FPS at every resolution, and alone renders 4096×4096 output in real time. For a 2048×4096 source panorama, the bake is a one-time 8.85 s, after which every frame renders at these rates. Figure 5 shows the quality gap at 4K, where both splatting pipelines render soft boundaries and blurred texture while MSI rendering approaches the ground truth with sharp boundaries and crisp texture.

4.4 Component Ablations

Table 3 ablates the reconstruction stage and renders directly from the predicted Gaussians, so the bridge does not mask the differences. Removing circular padding, which claims continuity across the $\pm 180^\circ$ seam and the poles, degrades both metrics the most. Removing sky-aware composition ranks second and leaves color casts at the sky-foliage boundary. Removing equal-area resampling changes the metrics least, since full-image averaging dilutes its visually prominent loss of sharpness around the equator. Appendix F.1 shows the rendered views behind these three rows. The lower group compares training with and without Stage 2, which fine-tunes on the pseudo novel-view training pairs of Tangent-View Scaffolding for in-the-wild generalization. Without it, the model scores best on the in-domain TartanGround scenes but worse on the real OmniPhotos captures, so the full model trades an in-domain cost for that generalization.

Figure 6 ablates the layer routing of Section 3.2, where hard and soft routing fail in complementary ways. Soft routing copies the flags in the red inset onto a nearer layer with no content of its own, and the copy shifts under parallax into a second flag. Hard routing leaves a dark streak along a layer interface in the blue inset and holes in the wall that the flags no longer cover. Hybrid routing renders both regions cleanly, since boundary-local softness removes the streak and fills the wall while the two gates block the duplicate.

5 Conclusion

We presented SphereBridge, a panorama-native pipeline for real-time high-resolution nearby-view rendering from a single panorama. A feed-forward model regresses a panorama-wide 3D Gaussian field in one pass, and a one-time bake converts it into an MSI whose layer textures carry the source panorama, so per-frame cost is independent of the Gaussian count and output detail is bounded by the panorama rather than the reconstruction. SphereBridge reaches state-of-the-art LPIPS and DISTS on indoor and outdoor benchmarks, including real captures, and renders 4096×4096 output at 23.92 FPS on one NVIDIA A800 GPU. Future work may extend the camera-motion range with generative priors for occluded content while keeping real-time multi-sphere rendering.

AI USE STATEMENT

In this work, we used generative AI tools (Claude, a large language model developed by Anthropic) to edit the manuscript for readability and to create and edit experimental code. We have not used generative AI tools to propose or refine hypotheses, to design or provide feedback on the research methodology or experiments, to implement the proposed method, or to analyze or interpret results, and the remaining tasks with required disclosure (synthetic data generation, theoretical models and mathematical claims or proofs, translation, dataset cleaning, and qualitative data analysis) are not applicable to this work. We have reviewed all AI-assisted work. All AI-assisted code was manually reviewed and tested for correctness by the authors, and all edited text was checked by the authors to preserve the intended meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *ECCV*, 2020.

Tobias Bertel, Mingze Yuan, Reuben Lindroos, and Christian Richardt. OmniPhotos: Casual 360° VR photography. *TOG*, 2020.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, 2017.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *CVPR*, 2024.

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

Zheng Chen, Chenming Wu, Zhelun Shen, Chen Zhao, Weicai Ye, Haocheng Feng, Errui Ding, and Song-Hai Zhang. Splatter-360: Generalizable 360 Gaussian splatting for wide-baseline panoramic images. In *CVPR*, 2025.

Shanding Diao, Yang Zhao, Yuan Chen, Zhao Zhang, Wei Jia, and Ronggang Wang. Multi-layer Gaussian splatting for single-image feed-forward spatial scene reconstruction. In *ACM MM*, 2025.

Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *TPAMI*, 2022.

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

Yijing Guo, Mengjun Chao, Luo Wang, Tianyang Zhao, Haizhao Dai, Yingliang Zhang, Jingyi Yu, and Yujiao Shi. PanoVGGT: Feed-forward 3D reconstruction from panoramic imagery. In *CVPR*, 2026.

Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-view view synthesis in the wild with learned adaptive multiplane images. In *SIGGRAPH*, 2022.

Huajian Huang, Changkun Liu, Yipeng Zhu, Hui Cheng, Tristan Braud, and Sai-Kit Yeung. 360Loc: A dataset and benchmark for omnidirectional visual localization with cross-device queries. In *CVPR*, 2024.

Zilong Huang, Jun He, Junyan Ye, Lihan Jiang, Weijia Li, Yiping Chen, and Ting Han. Scene4U: Hierarchical layered 3D scene reconstruction from single panoramic image for your immerse exploration. In *CVPR*, 2025.

Hualie Jiang, Ziyang Song, Zhiqiang Lou, Rui Xu, and Minglang Tan. Depth anything in 360°: Towards scale invariance in the wild. *arXiv preprint arXiv:2512.22819*, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *TOG*, 2023.

Numair Khan, Lei Xiao, and Douglas Lanman. Tiled multiplane images for practical 3D photography. In *ICCV*, 2023.

Xin Lin, Meixi Song, Dizhe Zhang, Wenxuan Lu, Haodong Li, Bo Du, Ming-Hsuan Yang, Truong Nguyen, and Lu Qi. Depth any panoramas: A foundation model for panoramic depth estimation. In *CVPR*, 2026.

Lars Mescheder, Wei Dong, Shiwei Li, Xuyang Bai, Marcel Santos, Peiyun Hu, Bruno Lecouat, Mingmin Zhen, Amaël Delaunoy, Tian Fang, Yanghai Tsin, Stephan R. Richter, and Vladlen Koltun. Sharp monocular view synthesis in less than a second. In *ICLR*, 2026.

Manthan Patel, Fan Yang, Yuheng Qiu, Cesar Cadena, Sebastian Scherer, Marco Hutter, and Wenshan Wang. TartanGround: A large-scale dataset for ground robot perception and navigation. In *IROS*, 2025.

Poly Haven. Poly Haven: The public 3D asset library. `https://polyhaven.com`, 2026.

Guo Pu, Yiming Zhao, and Zhouhui Lian. Pano2Room: Novel view synthesis from a single indoor panorama. In *SIGGRAPH Asia*, 2024.

Jiahui Ren, Mochu Xiang, Jiajun Zhu, and Yuchao Dai. PanoSplatt3R: Leveraging perspective pretraining for generalized unposed wide-baseline panorama reconstruction. In *ICCV*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michaël Ramamonjisoa, et al. DINOv3. *Transactions on Machine Learning Research*, 2026.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Yule Sun, Ang Lu, and Lu Yu. Weighted-to-spherically-uniform quality evaluation for omnidirectional video. *IEEE SPL*, 2017.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, Joao F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3D: Feed-forward generalisable 3D scene reconstruction from a single image. In *3DV*, 2025.

Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020.

Guangcong Wang, Peng Wang, Zhaoxi Chen, Wenping Wang, Chen Change Loy, and Ziwei Liu. PERF: Panoramic neural radiance field from a single panorama. *TPAMI*, 2024.

Dongyu Yan, Guanyu Huang, Fengyu Quan, and Haoyao Chen. MSI-NeRF: Linking omni-depth with view synthesis through multi-sphere image aided generalizable neural radiance field. In *WACV*, 2025.

Shuai Yang, Jing Tan, Mengchen Zhang, Tong Wu, Gordon Wetzstein, Ziwei Liu, and Dahua Lin. LayerPano3D: Layered 3D panorama for hyper-immersive scene generation. In *SIGGRAPH*, 2025.

Cheng Zhang, Haofei Xu, Qianyi Wu, Camilo Cruz Gambardella, Dinh Phung, and Jianfei Cai. PanSplat: 4K panorama synthesis with feed-forward Gaussian splatting. In *CVPR*, 2025.

Hao Zhang, Mohamed El Banani, Jen-Hao Cheng, Paul Zhang, Yi Hua, Ben Mildenhall, Christoph Lassner, Narendra Ahuja, and Gengshan Yang. World Tracing: Generative pixel-aligned geometry beyond the visible. *arXiv preprint arXiv:2606.13652*, 2026.

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

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

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

A Evaluation Pair Selection

This section gives the full pair-selection protocol referenced in the Datasets paragraph of Section 4.1. The main text uses a single-panorama protocol: each method receives one source ERP panorama, and target views are selected by a parallax-angle filter capped at 20° using source-view depth. The protocol below makes the parallax-angle definition, the depth-based threshold, and the per-dataset depth source explicit, so the evaluation pairs are reproducible from the dataset releases alone.

A.1 Protocol overview

For each scene, we hold out one source ERP panorama and treat every remaining panoramic view in the same scene as a candidate target. A candidate is admitted as an evaluation target if its relative translation with respect to the source falls inside a worst-case parallax-angle bound of 20°, computed against the source-view depth as defined in Section A.2. This produces a per-scene set of (source, target) pairs that share a common source viewpoint, mirroring the deployment regime of near-viewpoint immersive browsing. The same per-scene pair set is reused across all baselines so that any method-level difference is not confounded by pair selection.

A.2 Parallax-angle definition and threshold

Let S and T denote the source and target camera centers, $\mathbf{t} = T - S$ their relative translation in the source frame, and X a 3D scene point along a source ray with depth d. The parallax angle of X between S and T satisfies, by the law of sines,

$$\sin\theta \;=\; \frac{\|\mathbf{t}\|\,\sin\angle PTS}{d} \;\le\; \frac{\|\mathbf{t}\|}{d},$$

with equality at $\angle PTS = 90^\circ$, i.e. when $\mathbf{t} \perp \overrightarrow{TP}$. The $\sin$ form therefore gives a tight worst-case bound on the parallax angle for any scene point at depth d from the source. A $\tan$ form would correspond to the configuration $\mathbf{t} \perp \overrightarrow{SP}$, which is sub-optimal and underestimates the true worst case.

We summarize the source-view geometry by $d_{1\%}$, the first-percentile of valid source-view depth, evaluated only on pixels where the dataset depth is finite and inside the dataset-specific valid range. The first percentile is preferred over the minimum because it suppresses isolated depth outliers while still anchoring the bound to the closest robust scene structure. A candidate target is admitted if and only if

$$\|\mathbf{t}\| \;\le\; \sin(20^\circ)\cdot d_{1\%} \;\approx\; 0.342\, d_{1\%},$$

which is exactly the worst-case bound above instantiated at the 20° cap and the $d_{1\%}$ depth proxy. This single inequality is the entire admission rule: there is no additional overlap test, view-direction test, or scene-class heuristic.

A.3 Pair-construction algorithm

Algorithm 1 states the full per-scene procedure. The algorithm is dataset-agnostic: it consumes a list of panoramic views with extrinsics and an accessor for source-view depth, and emits the admitted (source, target) pairs. For datasets that release multiple captures per scene, the source view is fixed once per scene and reused, so that the same source is used across baselines and across re-runs.

A.4 Per-dataset depth source

We instantiate the protocol on the six evaluation datasets used in Section 4 (Matterport3D (MP3D) (Chang et al., 2017), Replica (Straub et al., 2019), 360Loc (Huang et al., 2024), TartanGround (Patel et al., 2025), PanoCity (Guo et al., 2026), OmniPhotos (Bertel et al., 2020)). The depth source used to compute $d_{1\%}$ differs across datasets, so we make it explicit before applying the admission rule. For Matterport3D (MP3D), Replica, TartanGround, and PanoCity, the source-view depth is the dataset-released ground-truth ERP depth from the rendered or reconstructed mesh. For 360Loc, the dataset releases mesh-derived panoramic depth maps constructed by reconstructing a TSDF/VDB mesh from LiDAR point clouds and casting one ray per ERP pixel from the camera

ALGORITHM 1: Per-scene parallax-angle pair selection.

Input: Source view V_s with extrinsics E_s and dense depth D_s; candidate views $\{V_t\}$ with extrinsics $\{E_t\}$; threshold $\theta_{\max} = 20^\circ$.
Output: Set $\mathcal{P}$ of admitted (source, target) pairs.

$D_s^{\text{valid}} \leftarrow \{\, D_s[u, v] : D_s[u, v]$ is finite and inside the dataset-valid range $\}$;
$d_{1\%} \leftarrow$ first percentile of D_s^{valid};
$t_{\max} \leftarrow \sin(\theta_{\max}) \cdot d_{1\%}$;
$\mathcal{P} \leftarrow \emptyset$;
foreach *candidate view* V_t **do**
  $\mathbf{t} \leftarrow$ translation of E_t relative to E_s;
  if $\|\mathbf{t}\| \le t_{\max}$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(V_s, V_t)\}$;
  end
end
return $\mathcal{P}$;

center to the mesh surface, and the ray-mesh intersection distance is used as the source-view depth. For OmniPhotos, no real dense depth is provided, so we generate a reconstructed radial depth map from the dataset's scene-dependent proxy mesh: for each ERP pixel we cast a spherical ray from the camera center, query its nearest intersection with the proxy mesh, and take the intersection-to-center distance as the depth. This OmniPhotos depth is used only for pair-selection $d_{1\%}$ estimation. It is never used as geometric supervision, since it inherits the proxy mesh's coarse approximation of true scene geometry.

B Per-Dataset Metrics Dashboard

The main paper reports cubemap-face LPIPS (Zhang et al., 2018) and DISTS (Ding et al., 2022) as primary perceptual metrics in Section 4.1. This section provides a full per-dataset metrics dashboard with two protocol families per dataset. The cubemap-side family covers PSNR, SSIM, LPIPS, and DISTS. The ERP-side family covers WS-PSNR (Sun et al., 2017), SSIM, LPIPS, and DISTS. The cubemap-side LPIPS and DISTS columns match the primary metrics used in Section 4.1 of the main paper and are repeated here so that every per-dataset table is self-contained. We report the pixel-level distortion metrics (PSNR, SSIM, WS-PSNR) for completeness, but, following standard practice for texture-rich novel-view synthesis, we do not use them as primary fidelity indicators because pixel-level fidelity is dominated by minor misalignment rather than perceptual quality (Mescheder et al., 2026).

B.1 Computation protocol

The four cubemap-side metrics share a single face layout: each method renders the six cubemap faces directly at the target pose, the ground-truth ERP view is projected onto the same six faces, the metric is computed per face, and the six face values are averaged. This shared convention prevents differences across metrics from being confounded by differences in resampling or face layout.

The four ERP-side metrics are computed directly on the predicted and ground-truth equirectangular images, with no cubemap reprojection. WS-PSNR applies the standard spherical-area weighting per pixel and serves as a sanity check against the cubemap-side PSNR. ERP-side SSIM uses the same per-pixel SSIM formulation as the cubemap-side variant but without spherical reweighting, which is how a naive ERP-evaluation pipeline would treat distortion. ERP-side LPIPS and DISTS are computed on the ERP image as a whole, even though the underlying feature networks are trained on perspective-image statistics. We therefore expect ERP-side LPIPS and DISTS to be biased by polar stretching, and we report them as a contrast against the cubemap-side variants to make this bias visible.

Table 4: Per-dataset metrics dashboard on TartanGround (Patel et al., 2025). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	20.49	0.641	0.386	0.204	19.45	0.704	0.314	0.151
Pano2Room	19.64	0.618	0.402	0.200	18.63	0.679	0.331	0.143
Depth-Backproj	22.12	0.695	0.363	0.190	21.20	0.760	0.272	0.135
Tangent-View Scaffolding	**22.89**	**0.720**	0.312	0.167	**22.18**	**0.787**	0.230	0.113
SphereBridge	20.97	0.677	**0.283**	**0.123**	20.22	0.740	**0.227**	**0.074**

Table 5: Per-dataset metrics dashboard on PanoCity (Guo et al., 2026). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	18.39	0.557	0.412	0.223	17.34	0.646	0.306	0.152
Pano2Room	19.00	0.582	0.371	0.190	18.18	0.661	0.286	0.122
Depth-Backproj	20.97	0.647	0.368	0.218	20.63	0.726	0.281	0.151
Tangent-View Scaffolding	**20.99**	**0.657**	0.374	0.221	**20.79**	**0.741**	0.279	0.154
SphereBridge	20.72	0.654	**0.237**	**0.107**	19.43	0.716	**0.186**	**0.060**

Table 6: Per-dataset metrics dashboard on OmniPhotos (Bertel et al., 2020). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	19.46	0.613	0.398	0.156	18.84	0.631	0.362	0.107
Pano2Room	18.85	0.602	0.408	0.153	18.32	0.617	0.376	0.105
Depth-Backproj	21.36	0.680	0.402	0.195	20.10	0.694	0.355	0.140
Tangent-View Scaffolding	**22.44**	**0.717**	0.359	0.169	**21.16**	**0.733**	0.323	0.121
SphereBridge	21.10	0.688	**0.317**	**0.094**	19.75	0.664	**0.294**	**0.064**

B.2 Per-dataset tables

Tables 4–9 report the full metrics dashboard for each of the six evaluation datasets, one table per dataset. Each table lists every method as a row and every metric as a column, with eight metric columns grouped into a cubemap-side block (PSNR, SSIM, LPIPS, DISTS) and an ERP-side block (WS-PSNR, SSIM, LPIPS, DISTS). This per-dataset layout keeps each table within a comfortable column count and makes within-dataset cross-metric trends easy to read horizontally across the two protocol blocks. All eight columns of a given table are computed on the same evaluation pairs and the same predicted and ground-truth panoramas as the primary metrics, so column-to-column differences reflect only the metric formulation and projection convention.

C Implementation and Training Details

The main paper trains SphereBridge on TartanGround (Patel et al., 2025) synthetic data and then fine-tunes it on self-captured unlabeled panoramas with tangent-view scaffolded inputs (Section 3.3 of the main paper). This section gives the architecture, optimization, loss formulation, inference, and

Table 7: Per-dataset metrics dashboard on Matterport3D (MP3D) (Chang et al., 2017). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	17.81	0.638	0.415	0.123	16.47	0.652	0.377	0.076
Pano2Room	24.92	0.811	0.237	**0.088**	23.38	**0.819**	0.214	**0.050**
Depth-Backproj	24.87	0.814	0.256	0.115	23.28	0.817	0.216	0.064
Tangent-View Scaffolding	**24.96**	**0.818**	**0.230**	0.096	**23.47**	**0.819**	**0.195**	**0.050**
SphereBridge	24.22	0.791	0.251	0.096	22.42	0.794	0.220	0.054

Table 8: Per-dataset metrics dashboard on Replica (Straub et al., 2019). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	**29.73**	**0.894**	0.198	0.132	**28.14**	**0.917**	**0.146**	0.077
Pano2Room	28.65	0.874	**0.192**	0.114	27.66	0.890	0.154	0.062
Depth-Backproj	26.98	0.860	0.304	0.162	25.67	0.883	0.230	0.096
Tangent-View Scaffolding	26.82	0.871	0.240	0.144	25.54	0.894	0.189	0.082
SphereBridge	26.42	0.844	0.197	**0.085**	24.46	0.846	0.175	**0.046**

Table 9: Per-dataset metrics dashboard on 360Loc (Huang et al., 2024). Cubemap-side metrics share the cubemap reprojection used by the primary metrics; ERP-side metrics are computed directly on the equirectangular image. Cubemap-side LPIPS and DISTS match the corresponding cells of the main-paper Table 1. The best value in each column is in bold.

	Cubemap-side				ERP-side			
Method	PSNR↑	SSIM↑	LPIPS↓	DISTS↓	WS-PSNR↑	SSIM↑	LPIPS↓	DISTS↓
PERF	17.39	0.535	0.442	0.214	17.02	0.594	0.391	0.137
Pano2Room	17.41	0.535	0.409	0.188	16.97	0.590	0.370	0.114
Depth-Backproj	18.20	0.561	0.466	0.240	17.56	0.617	0.406	0.161
Tangent-View Scaffolding	**18.39**	**0.589**	0.429	0.225	**17.77**	**0.657**	0.377	0.146
SphereBridge	18.03	0.583	**0.351**	**0.111**	16.21	0.572	**0.337**	**0.073**

hardware details needed to reproduce that pipeline. The two training stages are described separately, since they differ in data source, input construction, and learning-rate schedule.

C.1 Architecture summary

The panoramic depth prior is a DAP monodepth model (Lin et al., 2026) with a DINOv3 ViT-L backbone (Siméoni et al., 2026) that predicts a two-layer metric depth map from a single ERP panorama at the input resolution stated in the main paper. The two-layer output (Flynn et al., 2019) is obtained by duplicating the last convolution layer of the depth decoder, so the head predicts two distance layers per ERP pixel. All encoder layers are frozen while the depth decoder remains trainable, following SHARP (Mescheder et al., 2026). The reconstruction model uses a SHARP-style two-layer feature decoder with 32-channel output, on top of the same DINOv3 ViT-L backbone, and a shallow skip encoder inside the same Gaussian decoder that extracts local skip features from a 9-channel conditioning tensor that stacks RGB (3 channels), per-pixel ray direction (3 channels), the two-layer predicted depth map (2 channels), and the soft sky mask (1 channel). The feature-decoder and skip-encoder streams are fused, and prediction heads split the fused feature map into per-Gaussian deltas for color, opacity, scale, rotation, inverse distance, and angular offset. A sky-aware Gaus-

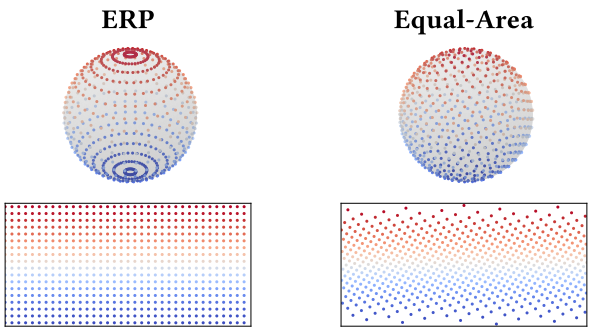

Figure 7: **Equal-area resampling.** ERP concentrates samples near the poles, while equal-area resampling distributes them uniformly on the sphere.

sian composer then applies the deltas. Each delta is added to the inverse-activated base value and the sum is passed through the attribute's activation: a sigmoid for linear-RGB color and opacity, a scale bound of 10, a unit quaternion for rotation, a safe-softplus for inverse distance with the delta scaled by 0.001, and a tangent-plane bound of 0.2 for the angular offset. The feature decoder, skip encoder, fusion module, and prediction heads together form the topology-aware Gaussian decoder of Section 3.1 of the main paper, the second of the two trained decoders. Per-Gaussian deltas are clamped to a log-delta range of ± 2. The two base Gaussians on a ray share the sampled RGB of that ray, take their scales from the layer depth and the angular spacing of the lattice, and start from default rotation and opacity. The initializer uses base opacity 0.5, base color in $[0.01, 0.99]$, and fixed sky distance 100. The sky mask is soft, a sigmoid of the predicted distance with slope 0.5 and threshold 90, so the sky branch is blended with the regular branch by this weight instead of being switched at a hard threshold. Every ERP-aligned input of the initializer and the decoder is resampled to the equal-area layout before use, namely the RGB, ray direction, two-layer depth map, and sky mask that form the conditioning tensor and initialize the base Gaussians, and the encoder features that enter the feature decoder. The initializer, the feature decoder, the skip encoder, and the fusion module therefore all operate in this layout, which replaces the polar over-sampling of the ERP grid with a uniform density on the sphere (Figure 7).

All convolutional modules of the topology-aware Gaussian decoder, that is the feature decoder, skip encoder, fusion module, and prediction heads, use the circular padding of Section 3.1 of the main paper, the circular wrap along longitude and the polar fold across the poles shown in Figure 3.

C.2 Tangent-view scaffold details

This section expands the tangent-view scaffold of Section 3.3 in the main paper. The scaffold first estimates a panoramic depth map for the source ERP with the pretrained panoramic depth model, which predicts panoramic disparity that is inverted and resampled to the source ERP resolution. A view sampler then cuts 18 overlapping tangent perspective views arranged as a horizontal ring of 8 views, a zenith and a nadir view, and two intermediate rings of 4 views above and 4 views below the horizon. For every view the sampler produces the tangent RGB crop, the projection of the panoramic depth map into that view, and the corresponding intrinsics and extrinsics, and each RGB-depth pair is resized to 1536×1536 before reconstruction.

Each view is reconstructed by the perspective single-view 3DGS regressor (Mescheder et al., 2026) with the projected depth supplied as its geometric prior, so all views inherit the metric scale of the shared panoramic depth. The projected depth is injected through SHARP's depth-adjustment module, which SHARP itself runs as an identity at inference, so the module's per-pixel scale map aligns each view's monocular depth with the projected depth. The per-view Gaussians are mapped into the shared world frame with the per-view camera poses. Because each view is reconstructed independently, a residual per-view scale error remains, and the scaffold aligns each view's Gaussians against the panoramic depth before fusion.

The fusion step resolves the overlap between neighboring views at the Gaussian level. Each Gaussian receives a view weight determined by where its center falls in the square crop of its tangent

view, so that the weight follows the crop boundary and Gaussians near the crop corners keep their contribution. With the center at (x, y, z) in the camera frame of the view, the crop distance $r = \max(|x|, |y|)/(z \tan(\mathrm{FoV}/2))$ is 0 at the view center and 1 on the crop boundary. The weight holds a plateau of 1 for $r \leq 0.6$, decays linearly with smoothstep smoothing for $0.6 < r < 1$, and is 0 for $r \geq 1$ or behind the camera, so view-central Gaussians dominate over view-boundary ones. The view weights are then normalized across the views that cover the same location, so a Gaussian covered by a single view keeps a weight of 1 and overlapping views share contribution in proportion to their weights. Gaussian centers are then binned into a uniform voxel grid with an edge length of 0.02 world units, and each voxel estimates a consensus depth as the view-weight-weighted mean depth of its Gaussians. Each Gaussian receives a consensus weight that falls from 1 at zero relative deviation from the consensus depth to 0 at a relative deviation of 15%, again as a linear ramp with smoothstep smoothing, and Gaussians in voxels covered by a single view keep a consensus weight of 1. The fused opacity multiplies the original opacity by the view and consensus weights.

C.3 Training stages and optimization

Shared supervision strategy. In Stage 1, each training sample is built around one source viewpoint and a set of additional viewpoints drawn by the view sampler, and all of them, source and novel, are projected to 512×512 cubemap faces before the rendering-supervision losses are computed. The novel viewpoints carry trajectory-defined rotation and translation, so the cubemap-face boundaries of different supervision views land at different angular positions of the source ERP and no single seam layout recurs across the training set. Combined with the seamless ERP conditioning tensor, this prevents cubemap-face seam discontinuities from being baked into the panorama-native reconstruction even though every per-view loss is evaluated on cubemap faces.

Stage 1 (synthetic pretraining). The model is trained on TartanGround panoramas with rendered ground-truth target views at the parallax-angle bound used for evaluation, sampling five non-adjacent views per training instance within a maximum inter-view distance of five frames along each trajectory. The TartanGround scenes used for evaluation in Section 4 are held out from this training split. Optimization uses Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay, and a peak learning rate of 1.6×10^{-4} on the reconstruction model. The unfrozen depth decoder is scaled by 0.01 so that its effective peak learning rate is 1.6×10^{-6}. The schedule is cosine annealing with a 5000-step linear warmup from $0.01\times$ the peak and a cosine minimum of 1.6×10^{-5}. The effective optimizer batch is 8 samples (2 GPUs $\times$ 2 per-GPU samples $\times$ 2 gradient-accumulation steps), trained for a total of 200000 steps with gradient clipping at norm 1. An exponential moving average with decay 0.999 is maintained from step 3000 onward, updated every 100 steps, and used for validation and downstream inference.

Stage 2 (scaffolded-input fine-tuning). The pretrained model is then fine-tuned from the Stage 1 checkpoint with input views generated by the tangent-view scaffold of Section 3.3 of the main paper. Fine-tuning data are drawn from a self-captured corpus of 3805 unlabeled panoramic videos, from which individual frames are sampled as source panoramas. These scaffolded views perturb the input, but they are not used as supervision targets. The training targets remain cubemap-face projections of the original ERP observations, so fine-tuning trains novel-view synthesis on real captures without injecting pseudo-label targets into the loss. Each training instance draws three non-adjacent frames at most three frames apart from one video and treats each as an independent source panorama with its own scaffolded input. Optimization keeps the Adam configuration of Stage 1 and the same peak learning rate of 1.6×10^{-4}, but uses a shorter cosine schedule with a 1000-step warmup and the same cosine minimum of 1.6×10^{-5}. The effective optimizer batch is 8 samples (2 GPUs $\times$ 4 per-GPU samples $\times$ 1 gradient-accumulation step), matching the Stage 1 effective batch, and Stage 2 runs for 60000 steps. The exponential moving average is disabled in Stage 2. The depth decoder is frozen throughout Stage 2, so the depth path inherited from Stage 1 is not updated on real captures. The scaffolded input is generated per training sample as follows. For each source ERP, the tangent-view scaffold (Appendix C.2) reconstructs a fused panoramic Gaussian scene, which serves as a stand-in 3D model of the scene. A perturbed camera pose is then sampled under the same parallax-angle bound used by the evaluation pair-selection protocol (Appendix A.2), with the depth proxy $d_{1\%}$ computed from the distances of the fused Gaussians to the source viewpoint in place of dataset-provided depth. Cubemap faces are rendered at the perturbed pose by Gaussian splatting

and reassembled into an ERP layout, which is used as the model input. The supervision targets remain the cubemap-face projections of the original captured panoramas at the actual viewpoints, so the input and the supervision targets never coincide. Stage 2 therefore feeds the model an always-scaffolded input distribution while keeping the supervision signal anchored to real captured pixels. The original ERP is never used as the input because that would collapse the loss into trivial source-view auto-encoding.

C.4 Loss formulation and weights

The total training loss contains seven terms grouped into rendering supervision, depth supervision, and regularization:

$$\begin{aligned}\mathcal{L} = \lambda_{\text{rgb}}\mathcal{L}_{\text{render_l1}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}} \\ + \lambda_{\text{depth}}\mathcal{L}_{\text{face_depth}} + \lambda_{\alpha}\mathcal{L}_{\text{alpha_bce}} + \lambda_{\text{erp}}\mathcal{L}_{\text{erp_depth}} \\ + \lambda_{\nabla d}\mathcal{L}_{\text{disp_grad}} + \lambda_{\text{sky}}\mathcal{L}_{\text{distant_mask_bce}}.\end{aligned}$$

The rendering-supervision group is computed on rendered cubemap faces. It includes an RGB L1 reconstruction term $\mathcal{L}_{\text{render_l1}}$, a VGG-19 perceptual term $\mathcal{L}_{\text{perc}}$, a cubemap-face depth L1 term $\mathcal{L}_{\text{face_depth}}$ that operates in inverse-depth space with an alpha-coverage gate of 0.1, and an alpha binary-cross-entropy term $\mathcal{L}_{\text{alpha_bce}}$. The depth-supervision group applies an ERP-area-weighted L1 loss $\mathcal{L}_{\text{erp_depth}}$ to the first predicted ERP depth layer of the depth-estimation module. The regularization group follows SHARP (Mescheder et al., 2026): the large-disparity gradient term $\mathcal{L}_{\text{disp_grad}}$ suppresses floaters with $\sigma = 0.01$ and $\epsilon = 0.01$, and $\mathcal{L}_{\text{distant_mask_bce}}$ adjusts the sky or distant-region mask with slope 0.5 and threshold 90. Stage 1 uses the full seven-term objective with weights $\lambda_{\text{rgb}} = 1.0$, $\lambda_{\text{perc}} = 0.002$, $\lambda_{\text{depth}} = 0.002$, $\lambda_{\alpha} = 0.02$, $\lambda_{\text{erp}} = 0.01$, $\lambda_{\nabla d} = 0.1$, and $\lambda_{\text{sky}} = 0.05$. Stage 2 drops the three terms tied to the frozen depth path, $\mathcal{L}_{\text{face_depth}}$, $\mathcal{L}_{\text{erp_depth}}$, and $\mathcal{L}_{\text{distant_mask_bce}}$, since reliable depth and sky-mask supervision is not available on the real captures used for fine-tuning. The four remaining terms keep their Stage 1 weights: $\lambda_{\text{rgb}} = 1.0$, $\lambda_{\text{perc}} = 0.002$, $\lambda_{\alpha} = 0.02$, and $\lambda_{\nabla d} = 0.1$.

C.5 Inference and rendering parameters

At inference, the model reconstructs panorama-native 3D Gaussians in one forward pass from a 512×1024 downsampled copy of the source panorama (matching the main paper). For the MSI rendering path, the predicted Gaussians are converted into a 16-layer multi-sphere image using the hybrid soft-hard routing, full-resolution layer compositing, and high-frequency texture injection of Section 3.2 of the main paper. After routing, the opacity of each Gaussian on each layer p is overridden to $\alpha_{p,t} = 1 - (1 - \alpha_t)^{w_{p,t}}$, written to the rasterizer through its logit, and every layer is rendered independently from the sphere center into an ERP texture. The remaining paragraphs of this subsection list the inference-time parameters of the MSI conversion.

Sphere-radius placement. The 16 spherical layers are placed adaptively per panorama. The near and far radii are set to the 1st and 99th percentiles of the reconstructed distance. The remaining radii are placed uniformly in inverse distance (i.e., in disparity) between the two endpoints, so that finer slabs cluster around closer surfaces where parallax is most visible. The same percentile rule is consistent with the $d_{1\%}$ depth proxy used by the evaluation pair-selection protocol (Appendix A.2).

Boundary-local softness. The geometric tent itself carries no free softness coefficient. The half-width of each layer's tent equals the half-width of its disparity slab, so the blend factor is zero at the slab center (pure hard routing) and one at the slab interface (pure soft routing). On top of the geometric tent the conversion applies an opacity-aware blend $s = s_b + w_o\, s_o\, (1 - s_b)$ with $s_o = (1 - \alpha)^{p_o}$ and default coefficients $w_o = 0.5$ and $p_o = 1.0$, which lets near-transparent Gaussians spill slightly more mass across boundaries while keeping near-opaque Gaussians close to hard routing.

Source-side foreground gate. The source gate prevents locally frontmost Gaussians from leaking soft mass into nearer layers where no occluding structure is available to absorb the spilled opacity. Each Gaussian is binned on an angular grid with 8×8 sphere pixels per bin at its home layer. The grid is dilated by a 5×5 kernel (circular along longitude, replicate along latitude), and a Gaussian

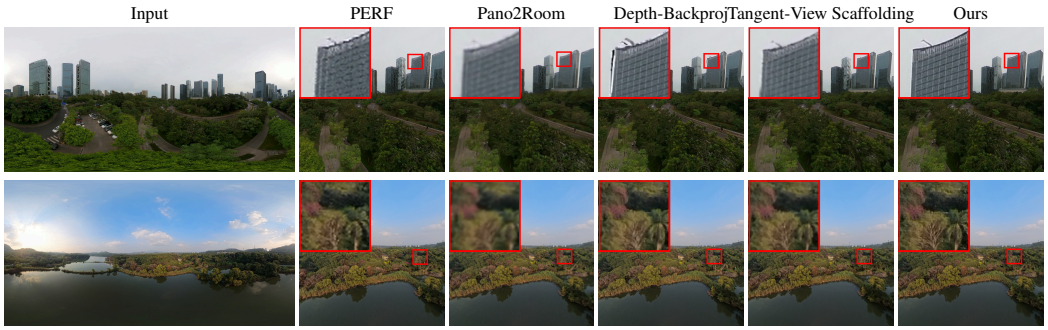

Figure 8: Qualitative novel-view synthesis on two in-the-wild panoramas we capture for this study, with the source panorama and outputs from four baselines alongside our method. Each rendered view carries a magnified inset of its red box. SphereBridge produces the highest perceptual quality.

is flagged as frontmost when the dilated count of occupied nearer layers in its 5×5 neighborhood is zero. Flagged Gaussians revert to hard routing.

Destination-side support gate. The support gate blocks off-home routes whose destination layer has no local angular structure. Reusing the dilated 5×5 occupancy from the source gate, an off-home route from a Gaussian to a destination layer q is admitted only if at least one other Gaussian in the 5×5 neighborhood has q as its home layer. Otherwise the soft mass to q is reassigned back to the home layer so the per-Gaussian layer weights remain stochastic. By default the support gate is applied only to nearer off-home routes, since those are the routes that produce ghosting in the absence of local support.

Alpha thresholding. No alpha drop threshold is applied in the conversion. A Gaussian contributes to a layer whenever its hybrid weight is non-zero, and the only numerical thresholds in the pipeline are a clamp to $[10^{-6}, 1 - 10^{-6}]$ before the per-layer logit transform and an $\epsilon = 10^{-6}$ in the alpha-premultiplied normalization, both of which serve numerical stability rather than contribution filtering.

Closed-form multi-sphere rendering. To render a target pixel from the baked MSI, the camera ray is transformed into the source coordinate system and intersected with each sphere of radius r_p in closed form. The nearest intersection direction determines the MSI texture coordinate. After sampling color $\mathbf{c}_p$ and opacity α_p on each layer, the renderer composites from near to far as

$$\mathbf{C} = \sum_{p=1}^{16} \left(\prod_{i=1}^{p-1} (1 - \alpha_i) \right) \alpha_p \mathbf{c}_p.$$

In the near-viewpoint regime the camera stays inside the MSI shells, so layer order follows sphere radius and rendering requires no per-frame sorting over scene primitives.

C.6 Hardware

Both training stages run on the same node of 2× NVIDIA A800 80GB GPUs under DDP, and inference and the rendering FPS measurements reported in the main paper run on a single NVIDIA A800 80GB from the same node. No cross-cluster mapping is needed: the FPS numbers in the main paper are produced on the same GPU model used for training.

D In-the-Wild Qualitative Results

Figure 8 compares SphereBridge with the four baselines of Section 4.1 on in-the-wild high-resolution panoramas captured for this study. These captures are texture-rich so that fine-detail preservation under high-resolution rendering can be inspected visually, and they come without

ALGORITHM 2: The Multi-Sphere Rendering Bridge bake (Section 3.2 of the main paper).

Input: Gaussian field $\{(\alpha_t, c_t)\}$ reconstructed from the 512×1024 downsampled copy of the source panorama I
Output: MSI of 16 RGBA layers $\{(c_p, \alpha_p)\}$ at the resolution of I
place radii $r_1 < \cdots < r_{16}$ uniformly in inverse distance between the 1st and 99th percentiles of the reconstructed distance;
foreach *Gaussian* t **do** `// hybrid soft-hard routing`
  set the home layer to the layer nearest in inverse distance;
  split weight between the home layer and its adjacent neighbor with the disparity-space tent filter;
  screen the neighbor-bound share with the foreground and support gates, returning blocked weight to the home layer $\Rightarrow w_{p,t}$;
end
foreach *Gaussian* t *and layer* p *with* $w_{p,t} > 0$ **do** `// optical-depth split`
  $\alpha_{p,t} \leftarrow 1 - (1 - \alpha_t)^{w_{p,t}}$;
end
foreach *layer* p **do** `// full-resolution layer compositing`
  rasterize the routed copies $\{(\alpha_{p,t}, c_t)\}$ of layer p from the sphere center into an ERP texture at the resolution of I, compositing front to back along every texel direction $\omega \Rightarrow \alpha_p(\omega),\ \bar{c}_p(\omega)$;
end
foreach *texel direction* ω *and layer* p **do** `// high-frequency texture injection`
  $v_p(\omega) \leftarrow \big(\prod_{q<p}(1 - \alpha_q(\omega))\big)\,\alpha_p(\omega)$;
  $c_p(\omega) \leftarrow v_p(\omega)\,I(\omega) + \big(1 - v_p(\omega)\big)\,\bar{c}_p(\omega)$;
end
return $\{(c_p, \alpha_p)\}_{p=1}^{16}$;

ground-truth novel views, so the comparison is qualitative. None of them overlaps the unlabeled panoramas used for fine-tuning. SphereBridge again produces the highest perceptual quality, and the supplementary material shows continuous nearby-view trajectories on these captures and on public panoramas from Poly Haven (2026).

E Source-View MSI Rendering Identity

This section gives the full derivation referenced in Section 3.2 of the main paper, showing that under our hybrid soft-hard routing, per-layer compositing, and the thin-slice approximation, MSI rendering at the source viewpoint reduces to a Gaussian-splatting rendering equation with the deterministic transmittance T_t replaced by an expected transmittance $\mathbb{E}[T_t]$ across the layers assigned to each Gaussian. This identity is what allows the rendering bridge to inherit the source-view appearance of the Gaussian field while still rendering through a layered spherical structure. Appendix E.5 then analyzes the source-view fidelity of the MSI after high-frequency texture injection. Algorithm 2 summarizes the full bake pipeline whose source-view behavior this section analyzes.

E.1 Setup and notation

Let $\{(\tau_t, c_t)\}_{t=1}^{N}$ denote the depth-sorted Gaussian field seen along a single source-view ray, where $\tau_t = -\log(1 - \alpha_t)$ is the optical-depth equivalent of the Gaussian's alpha α_t, and c_t is its color (Section 3.2 of the main paper). The MSI representation consists of $K = 16$ concentric spherical layers indexed by p. The hybrid soft-hard routing of Section 3.2 of the main paper gives each Gaussian a normalized weight vector $\{w_{p,t}\}_{p=1}^{K}$ over the layers, with $\sum_p w_{p,t} = 1$ for every t. Each layer composites its routed copies front to back (Section 3.2 of the main paper), and along the same ray the layer opacity and color reduce, under the thin-slice approximation $\alpha_{p,t} \approx w_{p,t}\,\tau_t$ introduced below, to the ray-level aggregates

$$\bar{\tau}_p \;=\; \sum_t w_{p,t}\,\tau_t, \qquad \bar{\tau}_p\,\bar{c}_p \;=\; \sum_t w_{p,t}\,\tau_t\,c_t,$$

with $\alpha_p = 1 - \exp(-\bar{\tau}_p)$ holding exactly because the transmittances of the routed copies multiply. These aggregates are evaluated at a single texel direction ω. Since this whole section works along one fixed source-view ray, the argument ω is dropped throughout.

E.2 GAUSSIAN-SPLATTING REFERENCE EQUATION

Front-to-back Gaussian-splatting compositing along the source ray gives

$$C_{\mathrm{GS}} \;=\; \sum_t T_t\,\alpha_t\,c_t, \qquad T_t \;=\; \prod_{a<t}(1-\alpha_a).$$

Substituting $\alpha_t = 1-\exp(-\tau_t)$ and applying the thin-slice approximation $\alpha_t \approx \tau_t$ that holds when the per-Gaussian optical depth is small, we obtain the τ-space reference form

$$C_{\mathrm{GS}} \;\approx\; \sum_t T_t\,\tau_t\,c_t, \qquad T_t \;=\; \exp\Big(-\sum_{a<t}\tau_a\Big).$$

This is the source-view rendering of the Gaussian field that Section 3.2 of the main paper refers to.

E.3 SOURCE-VIEW MSI RENDERING IDENTITY

The same front-to-back compositing rule applied to the layered representation produces, again under the thin-slice approximation,

$$C_{\mathrm{MSI}} \;\approx\; \sum_p T_p^{\mathrm{MSI}}\,\bar\tau_p\,\bar c_p, \qquad T_p^{\mathrm{MSI}} \;=\; \exp\Big(-\sum_{q<p}\bar\tau_q\Big),$$

where the layered transmittance is defined on the per-layer aggregates. Substituting $\bar\tau_p\,\bar c_p = \sum_t w_{p,t}\,\tau_t\,c_t$ and exchanging the order of summation gives

$$C_{\mathrm{MSI}} \;\approx\; \sum_p T_p^{\mathrm{MSI}} \sum_t w_{p,t}\,\tau_t\,c_t \;=\; \sum_t \Big(\sum_p w_{p,t}\,T_p^{\mathrm{MSI}}\Big)\tau_t\,c_t.$$

Defining the per-Gaussian expected transmittance as the layer-weight expectation of the layered transmittance,

$$\mathbb{E}[T_t] \;:=\; \sum_p w_{p,t}\,T_p^{\mathrm{MSI}},$$

yields the approximation described in Section 3.2 of the main paper,

$$C_{\mathrm{MSI}} \;\approx\; \sum_t \mathbb{E}[T_t]\,\tau_t\,c_t.$$

E.4 REDUCTION TO GAUSSIAN SPLATTING

Two limiting cases of the layer weights $w_{p,t}$ are useful for reading the identity. Under hard routing, where $w_{p,t} = \delta_{p=p(t)}$ pins each Gaussian to a single home layer $p(t)$, the expectation collapses to $\mathbb{E}[T_t] = T_{p(t)}^{\mathrm{MSI}}$, the transmittance of the home layer. Under soft routing across two adjacent layers, $\mathbb{E}[T_t]$ interpolates between the home-layer and neighbor-layer transmittances using the disparity-space tent weights of Section 3.2 of the main paper. In both cases, $\mathbb{E}[T_t] \to T_t$ whenever the per-layer accumulated optical depth up to a Gaussian's home layer matches the Gaussian-side accumulated optical depth $\sum_{a<t}\tau_a$, which holds when the layer ordering respects the depth ordering of the Gaussians and the assignment weights are concentrated on the correct layer. This is the regime in which source-view MSI rendering reproduces source-view Gaussian splatting up to the thin-slice approximation, and is the same regime under which the hybrid soft-hard routing of Section 3.2 of the main paper is designed to operate.

E.5 INJECTION AND SOURCE-VIEW FIDELITY

High-frequency texture injection (Section 3.2 of the main paper) updates each layer color as

$$c_p \;\leftarrow\; v_p\,I + (1-v_p)\,\bar c_p, \qquad v_p \;=\; \Big(\prod_{q<p}(1-\alpha_q)\Big)\alpha_p,$$

Table 10: Rendering quality versus camera displacement on TartanGround.

Displacement interval (m)	$(0, 0.25]$	$(0.25, 0.5]$	$(0.5, 1.0]$	$(1.0, 1.5]$
LPIPS↓	0.275	0.387	0.439	0.478
DISTS↓	0.122	0.152	0.188	0.220

where I is the source-panorama color along the ray and $\alpha_p = 1 - \exp(-\bar{\tau}_p)$ is the aggregated layer opacity. The visibility weights are the alpha-space form of the layered transmittance used above: since $1 - \alpha_q = \exp(-\bar{\tau}_q)$ holds exactly,

$$\prod_{q<p}(1-\alpha_q) \;=\; \exp\Big(-\sum_{q<p}\bar{\tau}_q\Big) \;=\; T_p^{\mathrm{MSI}}, \qquad v_p \;=\; T_p^{\mathrm{MSI}}\,\alpha_p,$$

so the two transmittance expressions coincide with no additional approximation.

Compositing the injected MSI at the source viewpoint applies the same visibility weights to the updated colors,

$$C_{\mathrm{inj}} \;=\; \sum_p v_p\, c_p \;=\; I \sum_p v_p^2 \;+\; \sum_p v_p\,(1-v_p)\,\bar{c}_p,$$

because the injection leaves all alphas fixed. When visibility concentrates on a single layer p^* ($v_{p^*} \to 1$, all other $v_p \to 0$), the composite reduces to $C_{\mathrm{inj}} = I$: wherever one layer dominates the source view, the injected MSI reproduces the source panorama exactly, independently of the thin-slice approximation used for the pre-injection identity. Where visibility is split across adjacent layers, near depth boundaries, $\sum_p v_p^2 < \sum_p v_p$ and the composite blends the source panorama with the aggregated low-frequency colors. High-frequency detail is therefore only partially injected inside these thin boundary bands, where its attribution to a single layer is inherently ambiguous. Occluded layers ($v_p \approx 0$) keep their aggregated colors, which is the disocclusion behavior described in Section 3.2 of the main paper.

F Additional Ablations

This section adds the qualitative counterpart of the reconstruction-side ablation of Section 4.4 and three studies of the rendering bridge, which vary the camera displacement, the number of MSI layers, and the layer placement. All studies run on TartanGround under the cubemap-face LPIPS and DISTS protocol of Section 4.1, with the reconstruction-side ablation rendered directly from the predicted Gaussians and the bridge studies through the baked MSI.

F.1 Reconstruction-side qualitative ablation

Figure 9 shows the rendered novel views behind the upper group of Table 3. Removing circular padding breaks continuity across both ERP boundaries, visible here at the $\pm 180^\circ$ seam, removing sky-aware composition leaves unnatural color casts at the sky-foliage boundary, and removing equal-area resampling loses sharpness around the equator, where the ERP grid is sparsest. The equatorial loss is visually prominent in the crops but small in Table 3, since full-image averaging dilutes it with the unaffected polar regions.

F.2 Camera-motion range

Table 10 reports rendering quality as a function of camera displacement. Instead of the parallax-angle filter of Appendix A, candidate targets are grouped into displacement intervals by their translation from the source viewpoint, and both metrics are averaged within each interval. Quality is best in the near-viewpoint interval and degrades monotonically as displacement grows, with no plateau. The degradation reflects the near-viewpoint regime of the MSI representation: the camera must stay inside the innermost sphere, which sits at the nearest reconstructed surface, and TartanGround cameras travel close to the ground, so the usable displacement range is short.

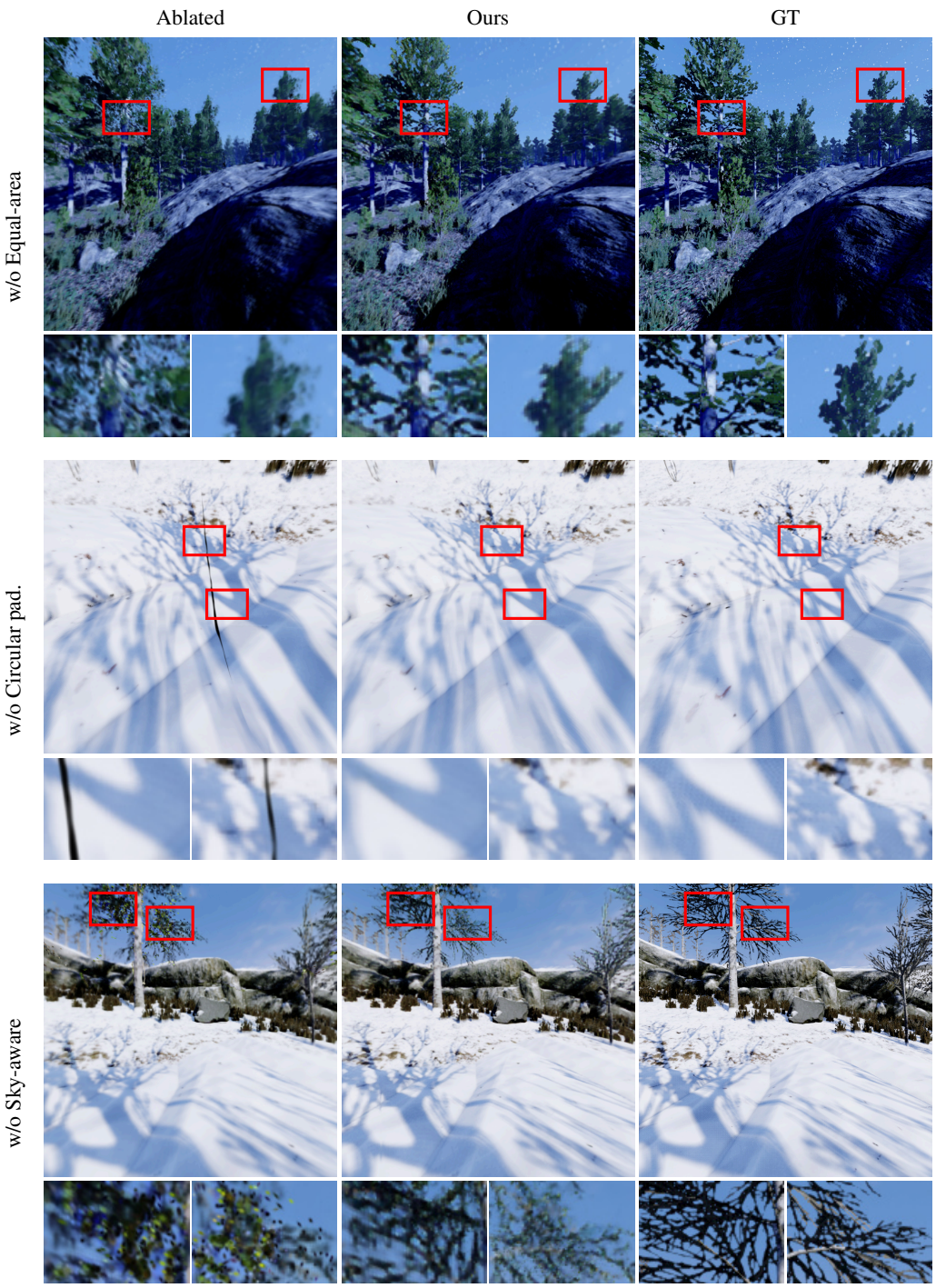

Figure 9: Qualitative reconstruction-side ablation on TartanGround, rendered directly from the predicted Gaussians. Each row removes one component of Section 3.1 and shows the ablated model, the full model, and the ground truth with two magnified crops.

F.3 Number of MSI layers

Table 11 varies the number of MSI layers K under the evaluation protocol above, so its $K = 16$ row coincides with the TartanGround entry of Table 1. Quality saturates around $K = 16$: moving from 8 to 16 layers improves LPIPS at unchanged DISTS, while moving on to 32 layers trades a marginal

Table 11: MSI layer-count ablation on TartanGround, with FPS measured on a single NVIDIA A800 GPU and the best value per column in bold.

			FPS↑	
K	LPIPS↓	DISTS↓	1024×1024	4096×4096
8	0.288	**0.123**	**1024.03**	**23.52**
16	0.283	**0.123**	926.61	22.96
32	**0.280**	0.126	719.66	22.27

Table 12: Layer-placement strategies at $K = 16$ on TartanGround, with the best value per column in bold.

Strategy	LPIPS↓	DISTS↓	Placement cost
Uniform (ours)	**0.283**	**0.123**	Negligible
k-means (Yang et al., 2025)	0.284	0.127	Seconds
Semantic (Huang et al., 2025)	0.285	0.125	Minutes

LPIPS gain for a DISTS loss. Rendering cost depends strongly on output resolution: increasing from 8 to 32 layers drops throughput from 1024 to 720 FPS at 1024×1024 output but only from 23.5 to 22.3 FPS at 4096×4096, since memory bandwidth and early ray termination dominate per-pixel cost at high resolution and mask the per-layer cost. At the target 4K output, additional layers are therefore nearly free, and we keep $K = 16$, where quality saturates.

F.4 Layer-placement strategy

Table 12 fixes $K = 16$ and compares the uniform inverse-distance placement of Section 3.2 against two alternatives from layered panoramic scene generation: k-means depth clustering following LayerPano3D (Yang et al., 2025) and semantic layering following Scene4U (Huang et al., 2025). The three strategies are nearly tied on both metrics, while their placement cost grows from negligible for uniform placement to seconds for clustering and minutes for semantic layering. With 16 layers the depth range is already sliced finely enough that placement has little room to change accuracy, so we keep the uniform placement as the cheapest strategy at no accuracy loss.