A. Method Details

Depth Estimation and Point Cloud Construction. We employ HunyuanWorld-Mirror [45] for metric depth estimation and camera parameter prediction from both single images and video frames. For the initial image I_0, the model predicts a metric depth map and camera intrinsics/extrinsics, which we back-project to obtain the initial point cloud $\mathcal{P}_0$. In subsequent iterations, we extract frames from generated videos and process each frame through HunyuanWorld-Mirror to obtain per-frame depth maps and camera parameters. To ensure geometric consistency, we apply multi-view geometric voting: for each candidate 3D point from frame k, we project it into all other frames and verify depth agreement within a tolerance of 0.1 meters. Points with support from at least 3 views are retained and assigned confidence scores based on the number of supporting views and local depth gradient magnitude (lower gradients indicate more reliable estimates). The filtered point clouds from all frames are merged with previous iterations' point clouds $\mathcal{P}_{t-1}$ through spatial binning at 5cm resolution, keeping points with highest confidence in each bin.

3D Scene Mesh Generation. We leverage Trellis [79], a SLAT-based 3D diffusion model, to generate complete scene meshes. The accumulated point cloud $\mathcal{P}_t$ is voxelized at resolution 128^3 to produce an occupancy grid $\mathcal{O}_t$ with three states: observed voxels (from point cloud), carved free-space (along viewing rays), and unknown occluded regions. For large scenes, we decompose the grid into overlapping 64^3 patches with 48-voxel overlap and extract corresponding image crops from accumulated observations. We employ Trellis's two-stage flow-matching generator: the sparse structure generator first completes the binary occupancy. The structured latent generator then produces per-voxel latent features conditioned on the completed occupancy and image crops. During generation, we apply test-time optimization with multi-view rendering: every denoising steps, we decode the current SLAT into 3D Gaussians, render from all accumulated views, and compute gradients of the photometric loss $\mathcal{L}_{\text{mv}}$ (Eq. 4) with weights $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$. These gradients guide 5 optimization steps (learning rate 1.0) before resuming diffusion. The final SLAT is decoded into both a mesh (via Marching Cubes) and 3D Gaussians for differentiable rendering.

Depth-Conditioned Video Generation. For novel view synthesis in Stage C, we render the completed mesh from $N = 121$ viewpoints along an orbital trajectory spanning the target rotation angle. Camera trajectories alternate between iterations: iteration 1 uses azimuth range $[0°, +45°]$ and iteration 2 uses $[0°, -45°]$, with elevation and radius fixed to the camera pose of the original reference image. For each viewpoint, we render a disparity map (inverse depth) from the mesh, normalize it to $[0, 1]$, and resize to 1024×1024. These disparity maps serve as geometric conditioning for Wan2.2-Fun-5B-Control [72], a depth-conditioned video diffusion model. We construct the prompt as: "Camera orbiting around a static 3D scene, filling the blank space with detailed and realistic details, geometry consistent, high quality, 8k. The video shows [caption]", where the optional caption is generated by Qwen3-VL-2B [80] from the input image. Video generation uses 50 DDIM sampling steps with classifier-free guidance scale 7.5, and the conditioning strength (controlnet scale) is set to 0.4 to balance geometric fidelity and visual quality. The input image I_0 is injected as the first frame to ensure appearance consistency.

Implementation and Computational Cost. All experiments are conducted on a system with one NVIDIA H100 (80GB) GPUs. Each pipeline iteration takes approximately 8 minutes: depth estimation and point cloud construction ($\sim$ 1 min), mesh generation with test-time optimization ($\sim$ 3 min), and video generation ($\sim$ 3 min). The complete three-iteration pipeline requires approximately 20 minutes per scene (No video generation for the last iteration). Peak GPU memory usage is approximately 68GB during mesh generation (for batch processing of 64^3 patches) and 45GB during video generation. All models are used off-the-shelf without fine-tuning, demonstrating the training-free nature of our approach.

B. More Experiment Results

Evaluation Metrics We employ both human evaluation and GPT-4o-based automatic evaluation to comprehensively assess 3D scene generation quality across three key dimensions: geometric quality, layout coherence, and texture consistency. Given a reference image and observations of two generated scenes (A and B), we ask annotators (or GPT-4o) to answer three questions: *(i)* Which scene has geometry that is more detailed, precise, and closer to the reference image? *(ii)* Which scene demonstrates a spatial layout and arrangement of objects that is more coherent and closely aligned with the layout in the reference image? *(iii)* Which scene exhibits textures that are significantly more coherent and consistent with the reference image? For human evaluation, annotators select either A or B for each question, and we compute win rate as the percentage of times our method is preferred. For GPT-4o-based evaluation, we prompt the model to directly return the answer and extract top-5 token log probabilities, obtaining $P(\text{A})$ and $P(\text{B})$ (set to 0 if not in top-5). When our method is option A, we treat $P(\text{A})$ as a soft vote; when our method is option B, we use $P(\text{B})$. The weighted win rate is computed as $P(\text{win})$ if our method wins, and $1 - P(\text{lose})$ if it loses, then averaged

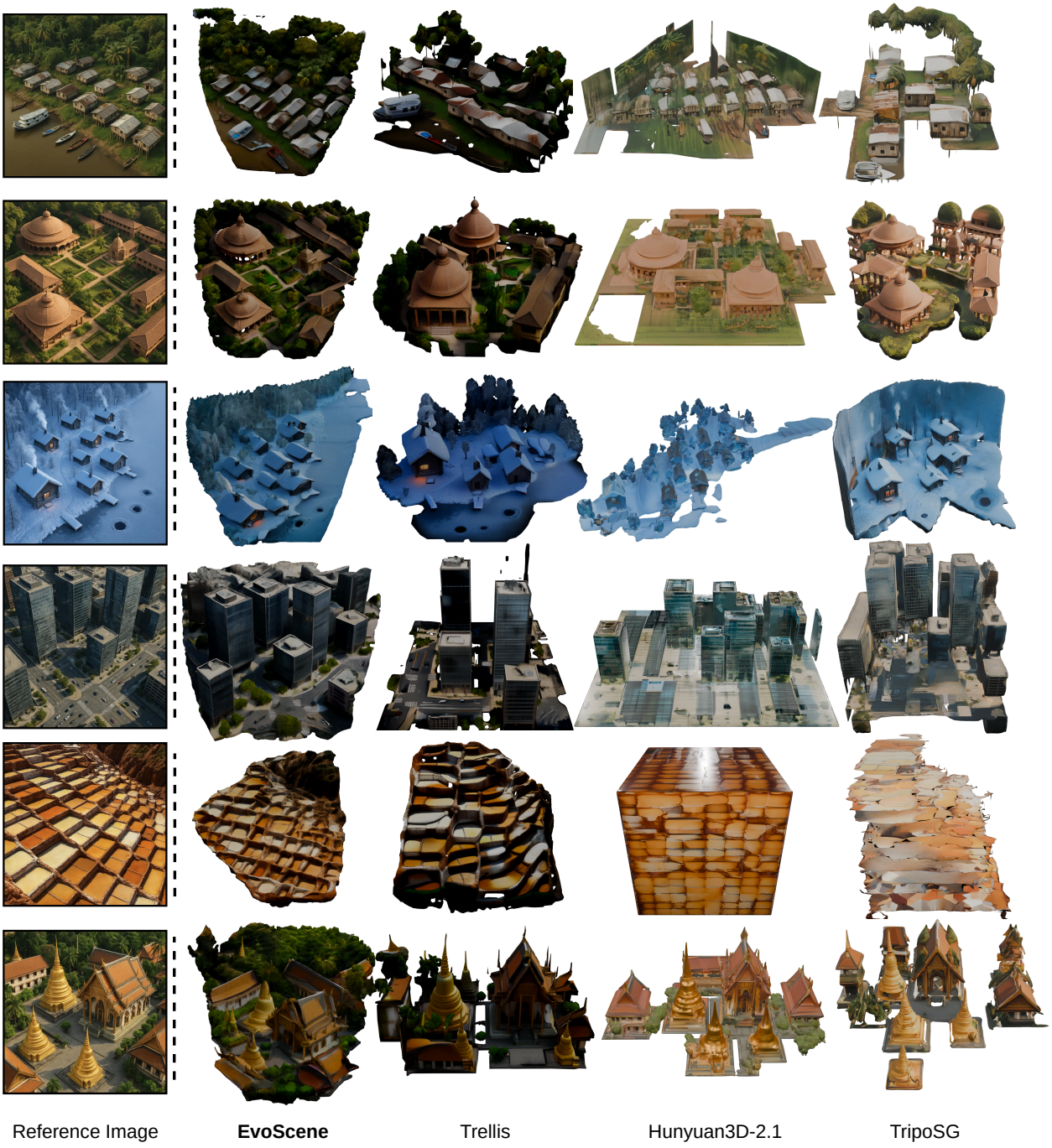

Figure 6. **Additional qualitative comparisons across diverse scene types.** From left to right: reference images, EvoScene (ours), Trellis, Hunyuan3D-2.1, and TripoSG. EvoScene produces complete geometry with preserved architectural details and photorealistic textures, while baselines exhibit fragmentation, flattened representations, or severe geometric distortions.

across all comparisons to capture both preference frequency and confidence strength.

Extended Baseline Comparisons Figure 6 presents additional qualitative comparisons between EvoScene and three strong baselines (Trellis [79], Hunyuan3D-2.1 [70], and Tri-

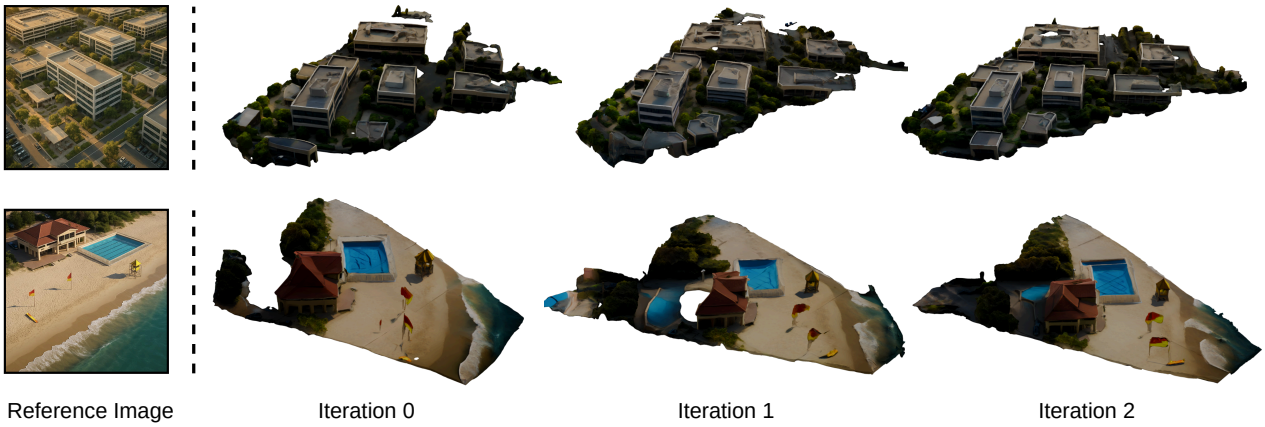

Figure 7. **Qualitative comparisons across different iterations.** The image shows the mesh differences between different iterations. With the iteration increasing, EvoScene can progressively improve the geometry and appearance quality, especially for unseen regions within the reference images.

poSG [35]) across diverse scene categories including urban environments, natural landscapes, architectural landmarks, and residential areas. As discussed in the main paper (Section 4.2), EvoScene consistently produces complete geometry with preserved architectural details and photorealistic textures, while baselines exhibit fragmentation (TripoSG), flattened representations (Hunyuan3D-2.1), or incomplete geometry with missing regions (Trellis). These additional examples further validate our method's superior performance across varying scene complexity and demonstrate robust generalization to diverse visual content.

Iterative Refinement Visualization Figure 7 illustrates the progressive improvement of geometry and appearance through our three-iteration self-evolving process (iterations 0, 1, 2). The initial iteration 0 reconstruction from only the input image produces reasonable but incomplete meshes with geometric errors in occluded regions and limited texture detail. Iteration 1 adds synthesized views from azimuth range $[0°, +45°]$, revealing previously occluded regions and improving background structure accuracy. Iteration 2 synthesizes complementary views from $[0°, -45°]$, achieving near-complete coverage with substantially refined geometric accuracy and texture consistency across the entire scene. The progression clearly demonstrates how our iterative geometry-appearance co-evolution progressively corrects initial errors, expands spatial coverage, and enhances both structural accuracy and visual realism, with the largest improvements occurring between iterations 0 and 1 when multi-view observations dramatically expand scene knowledge.