GRASE: GEOMETRY-NATIVE REAL-TO-SIM AGENTIC SCENE MODELING FROM VIDEO

Anonymous authors
Paper under double-blind review



Figure 1: GRASE transforms an input video into an editable 3D scene, recovering room structure, object layout, and fine-grained details. The example shows the progression from video observations to reconstruction, together with scene overviews, a close-up, and multiview renderings.

ABSTRACT

Real-to-sim conversion aims to turn a captured video of a real environment into an editable 3D scene while preserving the geometry and layout of what was observed. Existing methods often follow a one-pass, loosely coupled paradigm: each stage only take the limited feedbacks from previous stages, allowing errors to propagate without reliable correction and leaving the final scene inconsistent. We present GRASE, an agentic framework that closes this loop by treating the captured multi-view video observations as a standing reference throughout scene modeling. Instead of committing to intermediate estimation, GRASE revises its current hypothesis whenever rendered evidence disagrees with the capture, overcoming the fixed intermediate assumptions of conventional pipelines. Specifically, GRASE first fits the architectural shell and its openings from depth and multi-view cues. An agent then places generated assets in a shared coordinate frame while checking their spatial relationships accordingly. Finally each asset's geometry and placement are refined iteratively to improve agreement across the observed views. By coupling these steps through feedback, GRASE localizes structural discrepancies and revises only the responsible components, while preserving cross-view consistency. Under this pipeline, the resulting 3D scenes are faithful to the input, consistent across views and suitable for simulations. In our experiments, GRASE substantially outperforms other state-of-the-art methods, including a direct real-to-sim conversion using GPT6-Astra.

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

Real-to-sim aims to convert real-world observations into simulation-ready 3D environments, providing a scalable way to build training and evaluation data for embodied agents. Video is particularly attractive because it is inexpensive, easy to capture, and naturally records scene appearance and spatial structure. A robust video-based real-to-sim pipeline could therefore greatly reduce the cost of constructing digital twins for robot learning. Real-to-sim requires more than visually plausible reconstruction: the resulting scene must contain accurate geometry, object-level structure, and physically meaningful spatial relationships. Video makes this especially challenging due to occlusions, incomplete viewpoints, motion blur, depth ambiguity, and weakly textured regions. Small geometric errors can further lead to severe simulation failures such as penetration, floating objects, or incorrect contacts.

Existing approaches typically combine scene reconstruction, object segmentation, 3D generation, and scene assembly. However, they often fail to recover large room surfaces, produce inaccurate object geometry from partial observations, or rely on generative models that hallucinate shapes inconsistent with the input video. These errors accumulate during scene composition, resulting in scenes that may look plausible but are geometrically inconsistent and unsuitable for simulation.

An agentic formulation is attractive because real-to-sim naturally requires iterative diagnosis, tool use, and error correction rather than a single feed-forward prediction. In addition, modern VLM with powerful spatial perception capability such as GPT6-Astra enables a new real-to-sim paradigm of perceive-then-reconstruct. However, we find that it is not trivial to apply agentic models such as GPT-6 to acquire accurate geometry and topology. On one hand, for approaches that only use multi-view images, VLMs struggle to provide accurate 3D geometries that can be measured in real-world metrics. On the other hand, if point clouds and multi-view camera parameters were estimated and provided, the preliminary experiments suggest even powerful agents such as GPT6-Astra struggle to reason directly over raw point clouds, frequently making incorrect judgments about geometry, scale, and spatial relationships. This suggests that simply giving 3D observations to an agent is insufficient.

Our key idea is to couple explicit geometric modeling with an agentic verification and repair loop, using the original video as a persistent reference throughout reconstruction. The agent represents the scene through room surfaces and openings, object transforms, and editable object parts, so that discrepancies can be addressed through targeted geometric or topological changes. This leads to two closely connected components: **room-surface topology and geometry refinement** and **object-centric layout registration and part-wise geometric refinement**, integrated with **agentic execution and verification**.

Complicated room surfaces such as walls, ceilings and floor, are traditionally difficult to reconstruct. Thanks to the power of a VLM with strong spatial reasoning capability, we propose a perceive-then-reconstruct reconstruction paradigm, that perceives the architectural topology first, and then use geometric primitives and parametric patches to fit regular and irregular surfaces. This reconstruction paradigm allows use to reliably restore accurate geometry and topology using multi-view depth observations and image boundaries and agentic spatial reasoning and editing. Noteably, this reconstruction paradigm significantly improves the reconstruction results. In the ablation study, we observe a strong performance drop if this reconstruction paradigm is not used.

With the room surfaces reconstructed, the agent then aims to reconstructs object-level geometries that alignes with the input video. Agentic models such as GPT6-Astra is good at coding and editing 3D objects using geometric primitives in an agentic loop. But they struggles to align with real-world delicate and complicated geometries. Meshes generated by 3D generation models such as Trellis.2, however, is hard to edit via coding. We propose to combine 3D generation models and a part-wise geometric refinement module by first segmenting the generated geometries via off-the-shelf segmentation models, then use agentic models to perform part-wise editing, verification and repairs, while combining geometric cues such as point clouds and image boundaries.

To summarize, our contributions are:

- We propose *GRASE*, an geometry-native agentic real-to-sim pipeline that accurately reconstructs the scene from the input video, providing correct and consistent topology recovery for room surfaces.

- We propose an object-centric layout registration method that uses agentic part-wise editing, which delivers refined geometries that better aligns to the input video and are compatible to 3D generation models.

- In the experiments, GRASE performs significantly better to its GPT6-Astra counterparts, with or without using 3D point cloud modality and 3D generation models, which demonstrates the strength of our proposed the perceive-then-reconstruct paradigm.

2 Related Works

Image- and Video-Conditioned Compositional Scene Reconstruction. Reconstructing editable 3D scenes requires recovering object geometry and spatial configuration. Early methods jointly modeled object shapes, poses, layouts, and semantic or instance structure (Nie et al., 2020; Zhang et al., 2021; Dahnert et al., 2021). Geometric foundation models subsequently enabled dense correspondence estimation and reconstruction from unposed images (Wang et al., 2024a; Leroy et al., 2024; Yang et al., 2025a; Wang et al., 2025), while spatial memory supports incremental integration of observations (Wang & Agapito, 2025). Complementing these geometric representations, compositional approaches combine object generation with scene alignment (Shan et al., 2026; Zhou et al., 2024; Yao et al., 2025). Recent text-to-3D models such as Block3D improve the efficiency of object asset synthesis through block-wise diffusion and confidence-guided token correction (Cui et al., 2026). Other works jointly model multiple objects through diffusion (Huang et al., 2025; Dahnert et al., 2024). Deep Prior Assembly uses completion and candidate alignment, whereas CAST incorporates point-cloud conditioning and physical correction. OVOW (Chen et al., 2026) further extends instance-level mesh reconstruction to dynamic video scenes. Our work focuses on static scene reconstruction from video, combining point-cloud placement constraints with visual feedback to refine object poses and revisit mismatched components. This connects observation-based geometry with iterative correction of an editable, object-level scene representation.

Agentic 3D Scene Generation and Iterative Refinement. Language-guided systems translate descriptions into layouts, asset selections, and structured scene specifications (Feng et al., 2023; Yang et al., 2024; Wen et al., 2023; Çelen et al., 2024). Architect and LayoutVLM strengthen spatial guidance through hierarchical image inpainting and differentiable layout optimization, respectively (Wang et al., 2024b; Sun et al., 2025). Feedback-based approaches revisit scene states using rendered observations, scene judgment, or reflective tool selection (Hu et al., 2024; Ling et al., 2026; Yang et al., 2026; Xia et al., 2026). Among them, SceneCraft generates and refines Blender programs, while SAGE combines MCP-based orchestration with visual and physical critics. SceneSmith (Pfaff et al., 2026) organizes hierarchical construction through designer, critic, and orchestrator agents. Spatially contextualized VLMs (Liu et al., 2025) further integrate semantic point clouds, scene hypergraphs, and Blender-based verification. These methods motivate combining planning, geometric grounding, and feedback. Our framework centers this process on agreement with an input video: VLM criticism guides Blender-based pose adjustment, followed by comparison and regeneration of mismatched objects. The emphasis is on connecting scene-level feedback to targeted corrections during reconstruction.

Spatial Reasoning and Geometric Grounding in VLMs. Spatial benchmarks reveal persistent challenges in visual perception, spatial measurement, and reasoning across views or video frames (Fu et al., 2024; Yang et al., 2025b; Zhang et al., 2026). SpatialVLM, SpatialBot, and SpatialRGPT address these challenges through spatial supervision, depth-aware inputs, and region-conditioned reasoning (Chen et al., 2024a; Cai et al., 2025; Cheng et al., 2024). SPAR (Zhang et al., 2026) further scales spatial training data using scenes with 3D ground truth. Complementary methods connect language models to explicit scene representations through point-cloud features, visual prompts, or object embeddings and identifiers (Hong et al., 2023; Chen et al., 2024b; Huang et al., 2023). SpatialPIN (Ma et al., 2024) instead incorporates 3D priors through prompting and tool interaction without additional training. LayoutVLM and spatially contextualized VLMs demonstrate how geometric representations and structured relations support scene modification (Sun et al., 2025; Liu et al., 2025). Following this principle, we use VLMs for semantic interpretation and corrective feedback, while point-cloud constraints anchor object placement to observed geometry. This couples spatial reasoning with executable pose correction in the reconstructed scene.

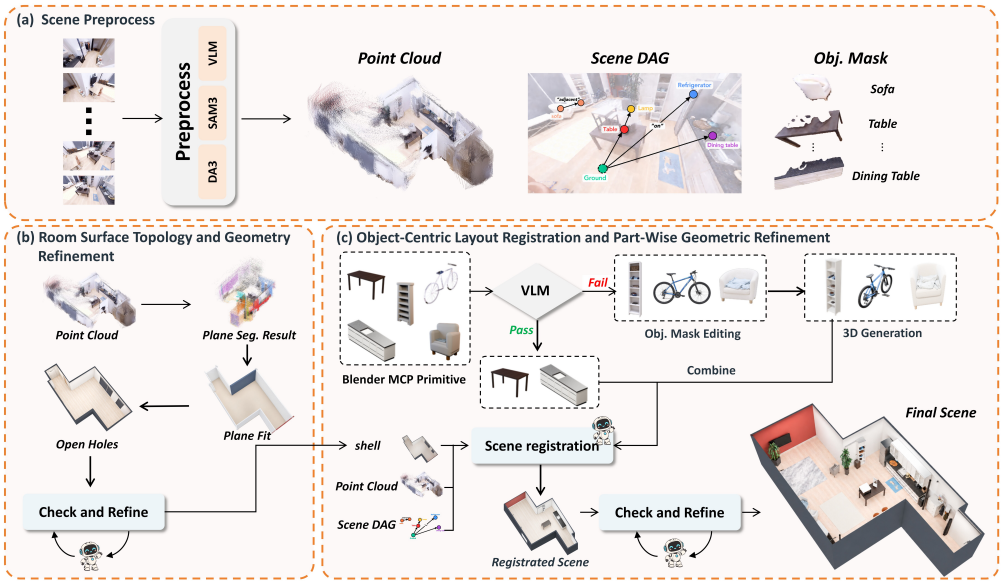

Figure 2: **Overview of GRASE.** (a) Scene preprocessing extracts a point cloud, a scene DAG, and object masks. (b) Room surface topology and geometry refinement reconstructs the shell and openings. (c) Object-centric layout registration and part-wise geometric refinement align object meshes and correct local shapes. Agentic verification guides both reconstruction stages.

3 Geometry-Native Agentic Real-to-Sim

Given a monocular video of a static indoor environment, our goal is to reconstruct an editable 3D scene $\mathcal{S}$ that is consistent with the observed views. The scene consists of room surfaces with explicit openings, one mesh per object with a world-space transform, and the support relations among surfaces and objects.

Figure 2 summarizes GRASE. Instead of letting a model produce geometry directly, GRASE exposes room surfaces, opening topology, object transforms, and object parts as variables that are fitted to geometric evidence and edited through constrained operations. After preprocessing, the shell is reconstructed together with its topology (Sec. 3.1), and objects are generated, registered, and corrected part by part (Sec. 3.2). Appendix A gives the implementation details, and Table 3 lists all constants.

Scene preprocessing. We uniformly sample T frames $\mathcal{I} = \{I_i\}_{i=1}^{T}$ from the video and process them jointly with Depth Anything 3 (DA3), which estimates per-frame depth z_i, per-pixel confidence, intrinsics $\mathbf{K}_i$, and camera-to-world poses $\mathbf{T}_i$. Each pixel $\mathbf{u}$ with valid, confident depth is lifted to $\mathbf{T}_i\big(z_i(\mathbf{u})\,\mathbf{K}_i^{-1}\tilde{\mathbf{u}}\big)$, where $\tilde{\mathbf{u}}$ is its homogeneous coordinate, and merging these points over all frames gives the scene point cloud.

Scene inventory. A GPT-6-based VLM organizes the frames into an inventory of surfaces and objects. A wall surface is a connected architectural face whose position and orientation vary continuously, so a smoothly curved wall is one surface, while sharp corners, structural steps, or spatial separation divide surfaces. By matching surfaces across views and inspecting their junctions, the VLM obtains the architectural surfaces $\mathcal{W}$ (walls, floor, and ceiling), their adjacency $\mathcal{A} \subseteq \mathcal{W} \times \mathcal{W}$ (surfaces meeting along a shared physical boundary), and the openings hosted by each surface. For objects, the VLM marks a 2D box around each visible object in every frame and groups the detections by physical identity using appearance, component structure, spatial context, and the recovered cameras. Each group is an *instance*, giving $\mathcal{O} = \{o_j\}_{j=1}^{N}$, where o_j has a category c_j and a 2D box in every frame of $\mathcal{V}_j$, the frames in which it is visible. The VLM further separates large, layout-defining objects (e.g., tables, sofas) from small movable ones (e.g., cups), which are generated and registered differently (Sec. 3.2), and infers a directed acyclic support graph $\mathcal{G} = (\mathcal{W} \cup \mathcal{O}, \mathcal{E})$ from

visible contact, relative position, and occlusion, where $(u, o_j) \in \mathcal{E}$ means that u supports o_j. Finally, SAM 3 segments every surface and object in its observed frames, and lifting the confident depth inside each eroded mask gives point clouds $\mathcal{P}_s$ for surfaces $s \in \mathcal{W}$ and $\mathcal{P}_j^{\text{obj}}$ for objects (Appendix A.1).

3.1 Room Surface Topology and Geometry Refinement

For each surface, the VLM decides whether a plane suffices or whether connected parametric patches are needed. Throughout, $\rho(x) = 2\big(\sqrt{1+x^2} - 1\big)$ denotes the soft-L_1 penalty, which limits the influence of outliers.

Regular surfaces. A planar surface s is represented as $\Pi_s = \{\mathbf{x} : \mathbf{n}_s^\top \mathbf{x} + d_s = 0\}$ with $\|\mathbf{n}_s\|_2 = 1$, initialized by RANSAC on $\mathcal{P}_s$ and refit to its inliers by SVD. The adjacency fixes the topology of the shell. If each wall were then cropped to its own points, neighboring walls would rarely meet exactly, leaving gaps or overlaps at the corners. We instead take a wall's extent from its neighbors. Three mutually adjacent surfaces s, t, r in $\mathcal{A}$, such as two walls and the floor, meet at the corner $\mathbf{v}_{str}$ that solves

$$[\mathbf{n}_s\ \mathbf{n}_t\ \mathbf{n}_r]^\top \mathbf{v}_{str} = -[d_s\ d_t\ d_r]^\top, \tag{1}$$

and these corners bound each wall as a finite polygon. Adjacent surfaces reference the same corners, which are recomputed whenever an incident plane moves. The adjacency therefore fixes the topology of the shell while the planes remain free to change: refining or editing one plane moves the shared corners with it and never opens a gap at a junction.

The initial fits are accurate in the middle of each surface but not necessarily at its junctions, the lines where it meets its neighbors, such as the corner between two walls or the base of a wall. To refine the planes, we fit the junctions observed in the images: notice that a junction is typically visible in multiple frames and each sighting of junction k in frame i is one observation $(i, k) \in \mathcal{J}$. For this observation, B_{ik} contains pixels sampled along the junction in frame i, away from occluders and image borders, and $\boldsymbol{\ell}_{ik}(\boldsymbol{\theta})$ is where the current planes place the junction in that frame: the projection of the two planes' intersection line, scaled so that $\boldsymbol{\ell}_{ik}^\top \tilde{\mathbf{u}}$ is a signed pixel distance. All planes are refined jointly by

$$\begin{aligned}\min_{\boldsymbol{\theta}}\ & \sum_s \frac{1}{|\mathcal{P}_s|} \sum_{\mathbf{p}\in\mathcal{P}_s} \rho\Big(\frac{\mathbf{n}_s^\top \mathbf{p} + d_s}{\sigma_s}\Big) + \frac{\lambda_{\text{jct}}}{|\mathcal{J}|} \sum_{(i,k)\in\mathcal{J}} \frac{1}{|B_{ik}|} \sum_{\mathbf{u}\in B_{ik}} \rho\Big(\frac{\boldsymbol{\ell}_{ik}(\boldsymbol{\theta})^\top \tilde{\mathbf{u}}}{\sigma_{\text{px}}}\Big) \\ & + \lambda_{\text{prior}} \sum_m \Big(\frac{\theta_m - \theta_{m,0}}{\epsilon_m}\Big)^2, \quad |\theta_m - \theta_{m,0}| \le \epsilon_m,\end{aligned} \tag{2}$$

where s ranges over the planar surfaces, $\boldsymbol{\theta}$ collects the plane offsets and normal corrections, σ_s and σ_{px} are fixed 3D and pixel scales, λ_{jct} and λ_{prior} are fixed weights, and ϵ_m bounds the change of each parameter from its initial value $\theta_{m,0}$. We solve this bounded robust least-squares problem with the cameras and $\mathcal{A}$ fixed (Appendix A.2). Because the shared corners are recomputed from the planes at every evaluation, the shell stays closed throughout the optimization. The resulting polygons are finally triangulated into editable Blender meshes.

Irregular surfaces. A surface that no single plane describes, such as a curved wall, is represented by connected patches, each mapping local coordinates (ξ, η) to a 3D point,

$$\mathbf{f}(\xi, \eta) = \mathbf{C}\,\mathbf{b}(\xi, \eta), \tag{3}$$

where $\mathbf{b}$ contains fixed basis functions and $\mathbf{C}$ their 3D coefficients. We use two families of patches. A *periodic loft* covers a region that wraps around the room; its coordinates are the horizontal angle and the height, and its horizontal position combines periodic trigonometric terms with quadratic terms in height. A *local quadratic displacement field* offsets a reference plane along its normal by a quadratic function, and a vertical reference normal makes it a height field. With the reference frame fixed, both families are linear in $\mathbf{C}$ and are fitted by least squares, with an envelope prior for poorly observed parts of a loft and stronger penalties on the quadratic terms of a displacement field to avoid unsupported curvature. The VLM specifies how neighboring patches meet. At a

sharp junction, we intersect the two surfaces and trim both parameter domains by the intersection curve, so the seam vertices coincide after triangulation. Where the transition is smooth, we blend the gradient-normalized implicit functions of the two patches with a smooth maximum and project nearby vertices onto the blend, whose width falls to zero near boundaries that must stay fixed. Please refer Appendix A.2 for details.

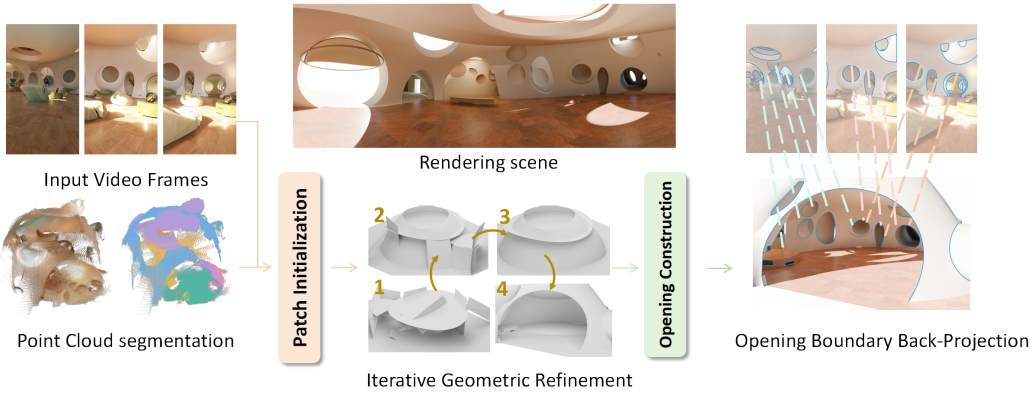

Figure 3: **Room-surface reconstruction and refinement.** Input video frames and the segmented point cloud guide patch initialization. The patches are iteratively refined using geometric evidence and comparisons between source frames and renderings. Observed opening boundaries are back-projected onto the fitted surfaces to construct explicit openings.

Openings and inferred geometry. Doors, windows, recesses, and passages are modeled as geometry rather than as appearance changes on an intact wall. For each inventoried opening, the VLM traces its visible rim in the source frames and determines whether it leads to a rear surface or an open passage. Rays through the traced rim are intersected with the host surface to initialize a 3D rim. We cut the enclosed region from the surface and build side surfaces along the rim, together with the rear surface when it is observed. Openings visible on a rear surface are built on that surface in turn, which handles nested structures. Occluded parts of the shell, such as the wall behind a curtain, are completed by extending the fitted surfaces under their parameterizations up to their junctions, and are marked as inferred rather than observed. The finished shell is rendered from the source cameras, and remaining discrepancies in surface shape, junctions, or opening rims go to the verification loop. Figure 3 illustrates this workflow.

3.2 Object-Centric Layout Registration and Part-Wise Geometric Refinement

With the shell in place, we build a mesh M_j and a world-space transform for every object o_j. The support graph $\mathcal{G}$ sets the order: large objects come first, and an object is placed only once everything supporting it is stable. Actual contact is checked against the supporting meshes during registration.

Object initialization Large objects, such as tables, chairs, and other furniture, are built directly in Blender by the VLM through the Model Context Protocol (MCP), using simple primitives for their major components. This gives clean, editable geometry whose parts follow the object's structure. When the VLM judges the resulting shape too poor, the object is generated with TRELLIS.2 instead, and small objects are always generated with TRELLIS.2 (Figure 4). The VLM builds each primitive model in place, using the location and extent of the object's point cloud $\mathcal{P}_j^{\text{obj}}$, whereas a TRELLIS.2 mesh arrives in its own canonical frame and is registered to $\mathcal{P}_j^{\text{obj}}$ by ICP. Since the point cloud covers only the visible side of the object and is noisy, either alignment can still be off in size or pose. The agent therefore selects three frames that together view the object from as many sides as possible, and the VLM compares renderings of the placed object with these frames and adjusts its size and pose accordingly. A detailed pipeline can be found in the Appendix A.3.

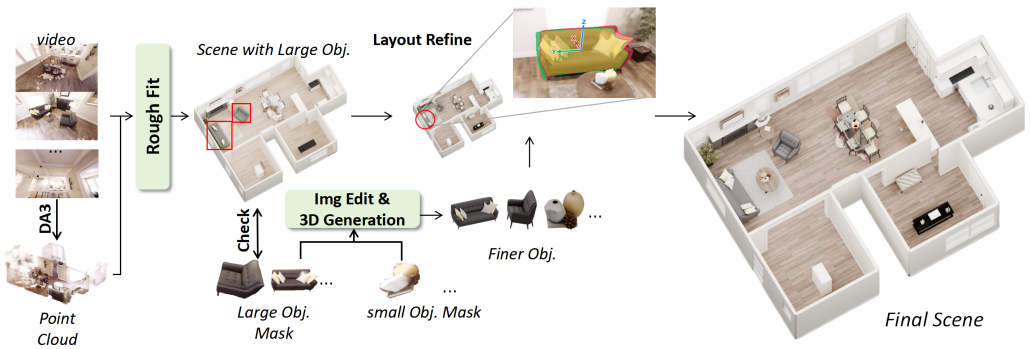

Figure 4: **Object generation and placement.** The large objects in "Scene with Large Obj." are built from primitives through Blender MCP. The VLM checks them against source views; only models with incorrect shapes or structures are replaced using TRELLIS.2. Small objects are generated directly with TRELLIS.2. All objects are then registered and refined to form the final scene.

Part-wise geometric refinement. A generated mesh may merge several components into one piece, such as a sofa and its cushions, and a single transform cannot fit them all: once the sofa body matches the observations, the cushions may still be misplaced. The VLM finds such a component by comparing renderings with the source frames, and the agent selects its region on the mesh by casting rays through the component's mask in several renderings. The component is then re-aligned to its own points while the rest of the mesh stays fixed. A component with wrong proportions, such as the sofa backrest in Figure 5, is stretched or contracted along a local axis $\mathbf{a}$ between a fixed anchor $\mathbf{x}_{\text{fix}}$ and a moving anchor $\mathbf{x}_{\text{mov}}$:

$$\mathbf{v}' = \mathbf{v} + S\big(\tau(\mathbf{v})\big)\,\Delta\,\mathbf{a}, \qquad \tau(\mathbf{v}) = \frac{(\mathbf{v}-\mathbf{x}_{\text{fix}})^\top \mathbf{a}}{(\mathbf{x}_{\text{mov}}-\mathbf{x}_{\text{fix}})^\top \mathbf{a}}, \tag{4}$$

for every vertex $\mathbf{v}$ of the component, where Δ is the signed length correction and $S(\tau) = 3\tau^2 - 2\tau^3$, with τ clamped to $[0, 1]$, is the cubic smoothstep, so the edit fades smoothly into the fixed geometry. An edit is kept only if the component matches the observations better without disturbing the rest of the object (Appendix A.4).

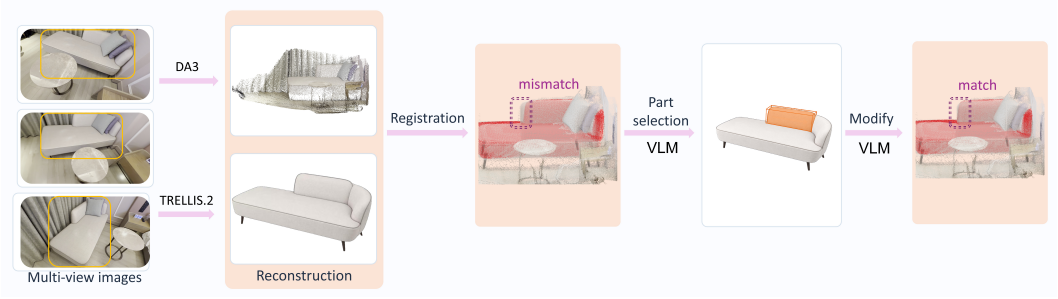

Figure 5: **Part-wise geometric refinement of a large object.** After object registration, a local mismatch remains in the sofa backrest. The VLM identifies the affected part and guides a constrained stretch that improves its alignment with the observed geometry while preserving the rest of the object.

Cross-object refinement. The previous steps place each object on its own, so relations between objects are not yet enforced: a computer may float above the desk it stands on, or two neighboring objects may interpenetrate. We therefore check every support relation in $\mathcal{G}$ against the actual geometry. An object is moved along the support normal until its base touches the supporting mesh, and samples across the contact region are checked for floating or penetration; wall-mounted and suspended objects are held to their observed attachment regions instead. Neighboring objects are

Table 1: Object-level and scene-level reconstruction and input-view rendering on Imaginarium and Replica. PC: point cloud. Imaginarium values are means over available scenes; object-level NC/CD exclude undefined values (8 and 13 valid scenes for the two GPT-6 baselines, respectively). Scene coverage, generation inputs, and alignment protocols differ across methods. – denotes unavailable rendering metrics for SAM3D, whose delivered GLB has no authored lighting or world setup. "w/o geo. critic" removes the joint point-cloud and hard-crease critic.

	Object-level				Scene-level			Input-view rendering		
Method	mIoU↑	NC↑	CD (cm)↓	F@5cm↑	CD (cm)↓	NC↑	F@5cm↑	PSNR↑	SSIM↑	LPIPS↓
Imaginarium										
SAM3D	0.0482	0.6182	7.1482	0.0648	72.1585	0.5892	0.1006	–	–	–
GPT-6 vanilla	0.1066	0.6490	18.4091	0.1144	18.1588	0.6844	0.3449	11.8223	0.5342	0.6481
GPT-6 + TRELLIS.2 + PC	0.2066	0.6850	11.8184	0.2062	10.9977	0.7946	0.5615	12.5762	0.5994	0.5606
GRASE (Ours)	**0.7425**	**0.7929**	**2.1936**	**0.8536**	**3.1412**	**0.9044**	**0.8866**	**15.5692**	**0.6667**	**0.3381**
Replica										
SAM3D	0.1566	0.6666	4.9177	0.1339	63.9258	0.6050	0.1908	–	–	–
GPT-6 vanilla	0.1897	0.6346	11.9005	0.1742	16.3887	0.6177	0.3452	11.1367	0.6943	0.5488
GPT-6 + TRELLIS.2 + PC	0.1325	0.5670	11.0108	0.1063	19.9107	0.6129	0.2165	10.8904	0.6373	0.5785
GRASE w/o geo. critic	0.1577	0.7076	9.7786	0.1740	9.9572	0.7729	0.5418	14.5470	0.6965	0.4190
GRASE (Ours)	**0.4121**	**0.7271**	**4.5207**	**0.4507**	**5.7963**	**0.8287**	**0.6477**	**16.3674**	**0.7454**	**0.3595**

Table 2: Architectural shell reconstruction. (a) Comparison on Imaginarium. (b) Ablation of the point-cloud and hard-crease critic, averaged equally over seven scenes after upright rigid alignment with fixed scale; CD is in 10^{-2} source-coordinate units and F-score uses a threshold of 0.05 units.

(a) Imaginarium

Method	CD (cm)↓	NC↑	F@5cm↑
GPT-6 vanilla	29.6426	0.7916	0.3474
GPT-6 + TRELLIS.2 + PC	6.5474	0.8785	0.6729
GRASE (Ours)	**2.9734**	**0.9548**	**0.8263**

(b) Critic ablation

Method	CD↓	NC↑	F-score↑
w/o Critic	55.74	0.859	0.134
Ours	**3.33**	**0.965**	**0.767**

also checked for interpenetration, and each correction is verified against the source frames through agent. Because contact is defined relative to the supporting mesh, objects are processed in the order of $\mathcal{G}$, and a later change to a support triggers new checks of the objects it supports (Appendix A.5).

4 Experiments

We evaluate GRASE in terms of architectural shell reconstruction, object-level and scene-level geometry, and input-view rendering quality. We compare against three baselines and assess the contribution of geometric feedback through ablation studies.

4.1 Experimental Setup

We evaluate GRASE on eight Replica and fifteen Imaginarium scenes selected by seeded random sampling against **SAM3D**, **GPT-6 vanilla** (direct scene construction from visual observations), and **GPT-6 + TRELLIS.2 + point cloud** (additionally using point-cloud/camera information and TRELLIS.2 asset generation). Both datasets share metric definitions: Chamfer Distance (CD), normal consistency (NC), and F-score at 5 cm (F@5cm) for geometry; instance mIoU for reconstructed–reference object agreement; and PSNR, SSIM, and LPIPS for input-view renderings against reference images. Architectural shells are evaluated separately from scene geometry.

4.2 Compare with baselines

Architectural shell reconstruction. On Imaginarium (Table 2a), GRASE achieves the best shell geometry, with a CD of 2.97 cm, an NC of 0.955, and an F@5cm of 0.826. Its CD is substantially lower than GPT-6 vanilla (29.64 cm) and GPT-6 + TRELLIS.2 + point cloud (6.55 cm), supporting the value of associating image observations with surface-specific 3D points and fitting connected shell geometry.

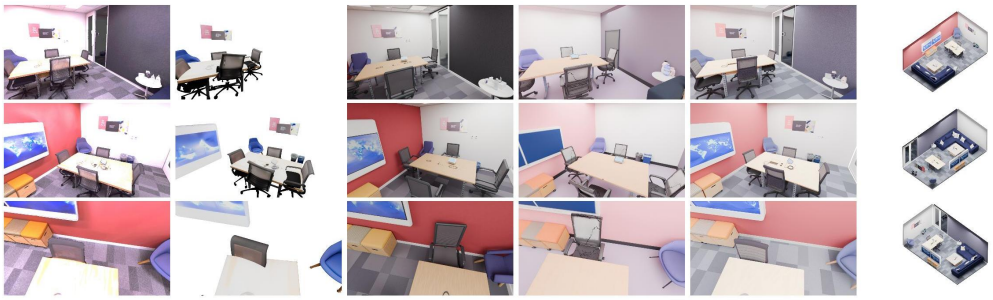

Figure 6: **Qualitative comparison on four scenes.** In every row, the input frame and the renderings of all methods share the same camera pose and intrinsics. The last column shows our reconstructed scene from a novel overview viewpoint. More examples can be found in Figure 7 in Appendix

Object and scene reconstruction. GRASE achieves the best object- and scene-level geometry and input-view rendering across both datasets (Table 1). GPT-6 vanilla captures semantic structure but struggles with geometry, scale, and placement. Adding TRELLIS.2 and point-cloud access improves Imaginarium rendering (PSNR: 11.82 to 12.58 dB; LPIPS: 0.648 to 0.561), but degrades both geometry and rendering on Replica. This inconsistency suggests that asset and point-cloud access alone is insufficient. GRASE identifies object support points through segmentation and depth back-projection, then explicitly aligns object scale and pose with this evidence.

Figure 6 compares SAM3D, GPT-6 vanilla, GPT-6 + TRELLIS.2 + point cloud, and our method on four scenes, with identical camera poses and intrinsics across methods within each row (see Figure 7 for more viewpoints).

Effect of the Geometric Critic. We remove the joint point-cloud and hard-crease critic, which evaluates candidate geometry against reliable point-cloud observations and multi-view hard-crease correspondences, while retaining the remaining agent workflow. For wall reconstruction (Table 2b), the full framework reduces CD from 55.74 to 3.33, a 94.0% reduction, while increasing NC from 0.859 to 0.965 and F-score from 0.134 to 0.767. On Replica (Table 1), the critic increases PSNR from 14.55 to 16.37 dB and SSIM from 0.697 to 0.745, while LPIPS decreases from 0.419 to 0.360. Removing geometric feedback thus degrades both wall accuracy and input-view rendering quality.

5 Conclusion, Limitations, and Future Work

We present GRASE, a geometry-native agentic framework that reconstructs editable indoor scenes from monocular videos through shell modeling, object layout registration, and part-wise refinement under multi-view verification, outperforming prior methods on Replica and Imaginarium. GRASE has several limitations. First, reconstruction remains limited in heavily occluded regions where the input video provides insufficient geometric evidence; such regions are completed by extending fitted surfaces or by generative priors and are marked as inferred rather than observed. Second, the scene inventory, support graph, and verification decisions rely on VLM judgments, so an early error, such as a missed junction or an incorrect support relation, can persist when the rendered evidence is ambiguous. Third, GRASE assumes a static scene and reconstructs geometry and layout only; articulation, materials, and physical properties required by many simulators are not yet recovered. Future work will extend the framework to articulated and dynamic objects, outdoor scenes, estimate physical and material properties from video, and replace part of the VLM-based verification with learned geometric critics to reduce cost and improve robustness.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper, specifically to improve clarity and concision. The authors reviewed all AI-assisted edits and take responsibility for the final content.

REFERENCES

Wenxiao Cai, Iaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. Spatialbot: Precise spatial understanding with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9490–9498. IEEE, 2025.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-design: Personalized llm interior designer. In *European Conference on Computer Vision*, pp. 217–234. Springer, 2024.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14455–14465, 2024a.

Junhao Chen, Boran Zhang, Mingjin Chen, Henghaofan Zhang, Saining Zhang, Congcong Zhu, Hao Zhao, Ruqi Huang, Zhihao Li, and Yufei Wang. One video, one world: Turning monocular video into physical 4d scenes. *arXiv preprint arXiv:2606.31388*, 2026.

Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. Ll3da: Visual interactive instruction tuning for omni-3d understanding, reasoning, and planning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26418–26428. IEEE, 2024b.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024.

Bowen Cui, Weijie Wang, Zeyu Zhang, Yefei He, Mingda Lin, Haoyu Zhao, Yuanyu He, Donny Y. Chen, Feng Chen, and Bohan Zhuang. Block3d: Efficient text-to-3d generation via block-wise diffusion. *arXiv preprint arXiv:2608.19567*, 2026.

Manuel Dahnert, Ji Hou, Matthias Nießner, and Angela Dai. Panoptic 3d scene reconstruction from a single rgb image. *Advances in Neural Information Processing Systems*, 34:8282–8293, 2021.

Manuel Dahnert, Angela Dai, Norman Müller, and Matthias Nießner. Coherent 3d scene diffusion from a single rgb image. *Advances in Neural Information Processing Systems*, 37:23435–23463, 2024.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in neural information processing systems*, 36: 18225–18250, 2023.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pp. 148–166. Springer, 2024.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Haifeng Huang, Yilun Chen, Zehan Wang, Rongjie Huang, Runsen Xu, Tai Wang, Luping Liu, Xize Cheng, Yang Zhao, Jiangmiao Pang, et al. Chat-scene: Bridging 3d scene and large language models with object identifiers. *arXiv preprint arXiv:2312.08168*, 2023.

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. Midi: Multi-instance diffusion for single image to 3d scene generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23646–23657. IEEE, 2025.

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

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Max Li. Scenethesis: A language and vision agentic framework for 3d scene generation. In *International Conference on Learning Representations*, volume 2026, pp. 136596–136629, 2026.

Xinhang Liu, Yu-Wing Tai, and Chi-Keung Tang. Agentic 3d scene generation with spatially contextualized vlms. *arXiv preprint arXiv:2505.20129*, 2025.

Chenyang Ma, Kai Lu, Ta-Ying Cheng, Niki Trigoni, and Andrew Markham. Spatialpin: Enhancing spatial reasoning capabilities of vision-language models through prompting and interacting 3d priors. *Advances in neural information processing systems*, 37:68803–68832, 2024.

Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 52–61. IEEE, 2020.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026.

Hui Shan, Keyang Luo, Ming Li, Sizhe Zheng, Yanwei Fu, Zhen Chen, and Xiangru Huang. Interact3d: Compositional 3d generation of interactive objects. *arXiv preprint arXiv:2603.16085*, 2026.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29469–29478. IEEE, 2025.

Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. In *2025 International Conference on 3D Vision (3DV)*, pp. 78–89. IEEE, 2025.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025.

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

Yian Wang, Xiaowen Qiu, Jiageng Liu, Zhehuan Chen, Jiting Cai, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Architect: Generating vivid and interactive 3d scenes with hierarchical 2d inpainting. *Advances in Neural Information Processing Systems*, 37:67575–67603, 2024b.

Zehao Wen, Zichen Liu, Srinath Sridhar, and Rao Fu. Anyhome: Open-vocabulary generation of structured and textured 3d homes. *arXiv e-prints*, pp. arXiv–2312, 2023.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21924–21935. IEEE, 2025a.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643. IEEE, 2025b.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *Advances in neural information processing systems*, 38:140319–140351, 2026.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16277–16287. IEEE, 2024.

Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. Cast: Component-aligned 3d scene reconstruction from an rgb image. *ACM Transactions on Graphics (TOG)*, 44(4):1–19, 2025.

Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3d scene understanding from a single image with implicit representation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8829–8838. IEEE, 2021.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Jilin Mei, Chunhui Chen, Yanpeng Zhou, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, et al. From flatland to space: Teaching vision-language models to perceive and reason in 3d. *Advances in Neural Information Processing Systems*, 38, 2026.

Junsheng Zhou, Yu-Shen Liu, and Zhizhong Han. Zero-shot scene reconstruction from single images with deep prior assembly. *Advances in Neural Information Processing Systems*, 37:39104–39127, 2024.

A Implementation Details of GRASE

This section gives the details omitted from Sec. 3. We keep the notation of the main paper, and Table 3 lists all constants.

A.1 Scene Preprocessing and Inventory

Depth and cameras. We sample T frames uniformly from each video and run DA3 on all of them jointly, obtaining depth maps, per-pixel confidence, intrinsics, and camera poses. Before back-projection, we check the camera and pixel-coordinate conventions of the DA3 outputs, discard invalid depth, and filter low-confidence pixels. A retained pixel $\mathbf{u} = (u, v)$ of frame i is back-projected as

$$\mathbf{x}_w = \mathbf{T}_i\big(z_i(\mathbf{u})\,\mathbf{K}_i^{-1}\tilde{\mathbf{u}}\big), \qquad \tilde{\mathbf{u}} = [u, v, 1]^\top, \tag{5}$$

where $z_i(\mathbf{u})$ is the camera-Z depth, $\mathbf{K}_i$ the intrinsic matrix, and $\mathbf{T}_i$ the rigid transformation from the camera frame of I_i to the world frame.

Segmented point clouds. For objects, each SAM 3 mask is eroded with a kernel of size r_{erode}. Inside the eroded mask, we keep valid, positive-depth pixels whose DA3 confidence exceeds $\max(\tau_{\text{conf}}, Q_\kappa)$, where τ_{conf} is a fixed floor and Q_κ is the κ-quantile of the confidences inside that mask. The retained pixels are subsampled to at most N_{view} per view and back-projected with Eq. (5). For surfaces, we likewise discard unreliable depth and points near segmentation boundaries, and we aggregate the points of each physical surface across frames using the cross-view correspondences in the inventory.

Scene inventory. In every sampled frame, the VLM places a 2D bounding box around the visible extent of each wall surface. The boxes localize surfaces for inspection only; they do not define boundaries or geometry. The VLM compares structural cues inside and around the boxes across successive and complementary frames to decide which observations show the same physical wall, accounting for viewpoint changes and partial occlusion, and it examines shared junctions across views to decide which surfaces are adjacent. Each visible opening is associated with its host surface. The surface inventory thus records cross-view correspondences, per-frame boxes and visibility, adjacency, and opening-to-surface relations.

Objects are handled in the same way. The VLM assigns a semantic category and a 2D box to every visible object and associates detections across views by consistency in appearance, component structure, and spatial context, using the recovered cameras to account for viewpoint changes. Repeated observations of one object share an identifier, while distinct objects of the same category keep separate identifiers. Large, layout-defining objects are distinguished from small ones by their observed extent and role in the scene, and support is inferred from visible contact, relative position, and occlusion cues across views.

A.2 Room Surface Reconstruction

Plane initialization. For each planar surface, we discard unreliable depth samples and points near segmentation boundaries. Every nondegenerate three-point sample of $\mathcal{P}_s$ defines a candidate plane, and RANSAC keeps the candidate supported by the most points within distance τ_{ransac}. We then refit the inliers by SVD of their centered coordinates, taking the right singular vector with the smallest singular value as $\mathbf{n}_s$ and setting $d_s = -\mathbf{n}_s^\top \bar{\mathbf{p}}$, where $\bar{\mathbf{p}}$ is the inlier centroid.

Corners and polygons. Every triple of mutually adjacent surfaces in $\mathcal{A}$ defines the corner

$$\mathbf{v}_{str} = -\begin{bmatrix} \mathbf{n}_s^\top \\ \mathbf{n}_t^\top \\ \mathbf{n}_r^\top \end{bmatrix}^{-1} \begin{bmatrix} d_s \\ d_t \\ d_r \end{bmatrix}. \tag{6}$$

Intersections with the adjacent walls, the floor, and the ceiling define each wall polygon, and neighboring polygons reference the same corner positions.

Junction term. The VLM marks visible physical junctions in the source frames and associates each with its two incident surfaces. We sample pixels along each observed junction, excluding foreground occlusions and image-border edges; for junction k in frame i, these pixels form B_{ik}. For the corresponding plane pair, we compute the 3D intersection line, project two of its points through camera i, and take the image line $\boldsymbol{\ell}_{ik}$ through the two projections, normalized so that its first two coefficients have unit Euclidean norm. The inner product $\boldsymbol{\ell}_{ik}^\top \tilde{\mathbf{u}}$ is then the signed distance in pixels from $\mathbf{u}$ to the line.

Optimization. The active parameters $\boldsymbol{\theta}$ are the plane offsets and corrections to the normal directions, and each may move by at most ϵ_m from its initial value. We solve Eq. (2) as a bounded robust least-squares problem with the cameras, observations, and adjacency fixed. At every function evaluation, the normals are renormalized and all shared corners are recomputed with Eq. (6). The fitted polygons are triangulated into Blender meshes and rendered from additional source views, where we check visible boundaries and point-to-triangle distances. A revision is retained only if it improves the targeted mismatch without opening gaps or degrading boundaries that were already aligned.

Occluded walls. For a partially occluded wall, the VLM matches the exposed wall regions across frames and identifies their junctions with neighboring walls, the floor, and the ceiling. We intersect these surfaces with the fitted wall plane and triangulate the resulting polygon across hidden regions, such as the part behind a curtain. The hidden part is marked as inferred, and the visible parts of the extended plane and its boundaries are checked against complementary source views.

Irregular patches. For an irregular surface, the VLM identifies visible surface regions and structural boundaries, which determine the support points and connections of each patch. We fit the coefficients $\mathbf{C}$ of Eq. (3) with the reference frames and prescribed coordinate components fixed;

planes, periodic lofts, and local quadratic displacement fields (including height fields) are the only representations used, and we do not search a larger primitive library.

Periodic lofts. The local coordinates are the horizontal angle and the height, and the horizontal position combines periodic trigonometric functions of the angle with quadratic terms in height. The coefficients are estimated by weighted linear least squares. To constrain poorly observed regions, we add samples from a prescribed elliptical envelope that narrows with height.

Local quadratic displacement fields. The reference origin is the coordinate-wise median of the support points. An initially near-vertical patch uses a horizontal reference normal pointing from the mean position of the cameras that observe it toward this origin. With the frame fixed, we fit the normal displacement by linear least squares, penalizing the quadratic coefficients more strongly than the others to limit unsupported curvature. Height fields, obtained with a vertical reference normal, are fitted with the soft-L_1 penalty ρ to reduce the influence of depth outliers. When a height field has an observed outer contour, we also adjust its coefficients and the parameters of an elliptical outline so that its projected boundary aligns with traced image curves, bounding and penalizing changes from the initialization.

Patch connections. For a sharp junction prescribed by the VLM, we bracket the roots of one surface's implicit equation along the other surface and solve for them with Brent's method. The resulting intersection curve trims both parameter domains. We triangulate the domains and evaluate the surface maps at their vertices; matching boundary samples give corresponding seam vertices identical positions in both patches. Where adjacent patches require a smooth connection, we locally subdivide mesh edges and clip overlapping portions. Each implicit surface function is divided by its gradient norm, and the resulting fields are combined with a smooth maximum. Nearby vertices are projected onto the zero level set of the blended field using finite-difference gradients and bounded steps. The blend width varies spatially and decreases to zero near protected boundaries, which keeps those boundaries fixed.

A.3 Object Registration

Primitive modeling. For a large object, the VLM assembles an editable model M_j^{prim} in Blender through MCP, using simple primitives for the observed major components; their dimensions and relative transforms encode the object's structure and proportions. The VLM places the model roughly: its position and size follow the center and extent of $\mathcal{P}_j^{\mathrm{obj}}$, its base rests on the supporting surface given by $\mathcal{G}$, and its facing direction, which the extent alone does not determine, is read from recognizable components in the frames, such as the side a chair's backrest is on. If the VLM judges that M_j^{prim} captures the observed components and proportions, it is kept as M_j.

Generation with TRELLIS.2. Otherwise, we pick three clear, complementary frames from $\mathcal{V}_j$, complete the partially occluded object in each by image editing, and generate one TRELLIS.2 candidate per completed image. Because TRELLIS.2 produces each mesh in its own canonical frame, we first align each candidate's local axes and contact reference and then place it using M_j^{prim} as the reference for position, orientation, and size. The candidates are rendered from the source cameras, and the VLM keeps the one whose components, proportions, and asymmetric structure best match the original frames; it replaces M_j^{prim} while keeping the instance's identity and support relations. Small objects skip primitive modeling: TRELLIS.2 generates their meshes directly, and the VLM initializes their position, scale, and orientation from the spatial distribution of $\mathcal{P}_j^{\mathrm{obj}}$ and the source images.

Pose parameterization. Registration keeps the mesh geometry and the cameras fixed and changes only the transform. For upright furniture, the support fixes the base height, roll, and pitch. Let $\hat{M}_j$ be the mesh divided by its local bounding-box dimensions, with its origin moved to the bottom center. In a frame whose vertical axis is the support normal, the transform is

$$\mathcal{T}_{\mathbf{q}}(\hat{\mathbf{x}}) = \mathbf{R}(\psi)\,\mathrm{diag}(\mathbf{e})\,\hat{\mathbf{x}} + \left[\mathbf{t}^\top,\, h_j\right]^\top, \quad \mathbf{q} = (\mathbf{t}, \psi, \mathbf{e}), \tag{7}$$

where $\mathbf{t} \in \mathbb{R}^2$ is the horizontal translation, $\mathbf{R}(\psi)$ the rotation by yaw ψ about the support normal, $\mathbf{e} \in \mathbb{R}^3_{>0}$ the dimensions along the object's local axes, and h_j the base height given by the support.

The admissible box $\mathcal{B}_j$ fixes an interval around the initial parameters $\mathbf{q}^{(0)}$ for every parameter: translation limits scale with the initial object extent, yaw has a bounded deviation, and dimension limits are proportional to the initial dimensions. For an upright object on a tabletop, $\mathbf{q}$ contains the two in-plane translations and the rotation about the tabletop normal, and scale is added only if the projected dimensions remain inconsistent after the pose is corrected. All distances are measured in the common reconstruction coordinate system.

ICP. We sample N_{surf} points on the surface of $\hat{M}_j$ and build a KD-tree over them. At iteration t, each observation $\mathbf{p}_n \in \mathcal{P}_j^{\text{obj}}$ is mapped into normalized model coordinates by $\mathcal{T}_{\mathbf{q}^{(t)}}^{-1}$ and assigned its nearest surface sample $\hat{\mathbf{y}}_n$. With these correspondences fixed, we solve

$$\mathbf{q}^{(t+1)} = \arg\min_{\mathbf{q}\in\mathcal{B}_j} \sum_n \rho_\delta\big(\mathcal{T}_{\mathbf{q}}(\hat{\mathbf{y}}_n) - \mathbf{p}_n\big) + \rho_\delta\big(w_{\text{prior}}(\mathbf{q} - \mathbf{q}^{(0)})\big), \tag{8}$$

where ρ_δ applies $\rho(\cdot/\delta)$ to every scalar component of its argument and sums the results, δ is a fixed soft-L_1 scale, and w_{prior} is a fixed prior weight. Each inner solve has a fixed budget of function evaluations. Correspondence assignment and parameter estimation alternate until the largest absolute parameter change falls below ϵ_{icp} or N_{icp} iterations are reached.

VLM-guided adjustment. The agent selects three frames from $\mathcal{V}_j$ that together view the object from as many sides as possible and renders the placed object from the same cameras. The VLM compares each rendering with its frame, aided by overlays that mark the area covered only by the rendering and the area covered only by the observation, and adjusts the size and pose within $\mathcal{B}_j$. Small objects occupy few pixels in distant views, so their adjustment uses close-up frames. Each instance is adjusted with its own observations, including when several instances share one generated mesh.

Silhouette refinement. The adjustment can be made precise with a silhouette objective over all views $i \in \mathcal{V}_j$. Let A_i be the observed instance mask, $H_i(\mathbf{q})$ the visible mask of the object rendered together with the rest of the scene, and $L_i^{\text{mask}} = 1 - \text{IoU}\big(A_i, H_i(\mathbf{q})\big)$. We minimize

$$L(\mathbf{q}) = \frac{1}{|\mathcal{V}_j|}\sum_{i\in\mathcal{V}_j}\big(L_i^{\text{mask}} + \lambda_{\text{edge}}E_i\big) + \lambda_{\text{worst}}\max_{i\in\mathcal{V}_j} L_i^{\text{mask}} + \lambda_{\text{pc}}L_{\text{pc}}, \tag{9}$$

$$E_i = \frac{\bar{d}(\partial H_i, \partial A_i) + \bar{d}(\partial A_i, \partial H_i)}{2\,\text{diag}(A_i)},$$

where λ_{edge}, λ_{worst}, and λ_{pc} are fixed weights, $\bar{d}(X, Y)$ is the mean distance from the pixels of X to the nearest pixel of Y, $\text{diag}(A_i)$ is the diagonal of the bounding box of A_i, and L_{pc} averages normalized and capped point-to-surface distances. The worst-view term keeps a single badly aligned view from disappearing into the average. To render $H_i(\mathbf{q})$, we ray-cast the target object and the rest of the scene from the recovered camera and keep the target pixels that are not occluded by other scene geometry, up to a numerical depth tolerance τ_{depth}. Masks are evaluated at a reduced resolution with correspondingly scaled intrinsics. The boundary ∂X of a mask X is obtained by subtracting its eroded version, and image-border pixels are excluded. Euclidean distance transforms give the mean nearest-boundary distances $\bar{d}$ in both directions; the normalizer $\text{diag}(A_i)$ is computed once at the reduced resolution, and an empty rendered boundary receives a fixed penalty E_{empty}. For L_{pc}, each world-space correspondence distance is divided by a distance scale d_{pc}, capped at τ_{cap}, and averaged over the observations; correspondences are recomputed with the same normalized-space nearest-neighbor rule as in ICP, using a separate surface sample of the mesh.

Coordinate search. Because L is built from binary masks and is not differentiable in $\mathbf{q}$, we minimize it by bounded coordinate search. The search starts from whichever of $\mathbf{q}^{(0)}$ and the ICP solution has the lower L. The translation, yaw, and lower dimension bounds of $\mathcal{B}_j$ are kept, while the upper dimension bounds are relaxed by a predefined amount β_{dim}. Translation and dimension steps are

proportional to the initial object dimensions, and yaw uses a fixed angular step. For each coordinate in turn, we evaluate a positive and a negative step, both clipped to the bounds, by re-rendering all visible masks and recomputing L, and we keep the better trial only if it reduces L by more than ϵ_{cs}. Step sizes decrease over N_{lvl} search levels, each with a budget of N_{sweep} sweeps, and a sweep without improvement advances the search to the next level.

Final check. The accepted transform is applied in Blender and re-rendered from the source cameras. We recompute point-to-surface distances against the full triangle mesh and inspect the overlays before retaining the placement.

A.4 Part-Wise Refinement

The VLM compares the source views with the current renderings to identify a component that the object-level transform cannot fit, for example cushions that remain misplaced once the sofa body matches, or an overly wide backrest whose seat and armrests already match. To select the component, the agent marks its visible regions in multiple renderings of the current object, and camera rays through these regions are intersected with the mesh to find the corresponding triangles. The selection is highlighted in complementary model views so that the agent can check that it covers the intended component and excludes neighboring parts, and additional views are used to add side and rear surfaces and to correct selection errors. The vertices of the reviewed selection form a vertex group $\mathcal{U}$; the mesh is neither cut nor split into a separate object. These rendered annotations only determine which existing geometry is editable; the source frames remain the reference for the target shape.

For a stretch, the fixed anchor $\mathbf{x}_{\mathrm{fix}}$ is placed where the component's alignment or attachment should be preserved, and the moving anchor $\mathbf{x}_{\mathrm{mov}}$ toward the extent that needs correction. Corresponding landmarks in the current renderings are ray-cast onto the mesh to locate both anchors in object-local coordinates. The agent estimates Δ from the source boundaries and the available reliable depth, and checks the proposed change in complementary source views before applying Eq. (4) to the vertices in $\mathcal{U}$. After each edit, we re-render from the relevant source cameras and assess the component's boundary alignment, its connection to neighboring components, and its support contact. The edit is kept only if it improves the targeted discrepancy without degrading other components or introducing severe interpenetration; otherwise, the previous geometry is restored.

A.5 Cross-Object Refinement

For each support relation $(u, o_j) \in \mathcal{E}$, the object is translated along the support normal until its base touches the mesh of u, which stays fixed. Several samples across the intended contact region are then checked so that the object neither floats above nor penetrates its support. Wall-mounted and suspended objects are constrained by their observed attachment regions instead. Each correction is also checked in complementary views and against neighboring objects for spacing and occlusion. Supports are processed before the objects they support; objects on different supports are processed independently, and a later change to a support requires re-checking the objects it supports.

A.6 Agent

The skill specification defines the stage dependencies, the tools available at each stage, and the criteria a candidate must meet to be accepted. Each review records the localized discrepancies and the corrective actions taken, and every accepted scene state is stored so that later revisions can be rolled back to it. Both reconstruction stages are executed by a tool-using VLM agent that follows a reusable skill specification defining the stage dependencies, available operations, and verification criteria (Appendix A.6). Rather than producing geometry itself, the agent works through tool calls: frame extraction, DA3 inference, image editing, TRELLIS.2 generation, and Blender operations through MCP. Numerical tools perform the shell fitting, object registration, mesh measurements, and mask comparisons, while the agent makes the decisions they cannot: which representation a surface needs, which frames to use, which component is wrong, and whether a result matches the video. Following the design–critique–orchestration organization of SceneSmith Pfaff et al. (2026), we describe this work in three roles.

Table 3: **Constants used by GRASE.** Symbols follow Sec. 3 and Sec. A.

Quantity	Symbol	Value
Scene preprocessing		
Frames sampled per video (real / simulated)	T	100 / 60
Mask erosion kernel	r_{erode}	3×3
Confidence floor / quantile	$\tau_{\text{conf}}, \kappa$	$1.2/0.35$
Points per view	N_{view}	650
Object generation		
Source views per large object	–	3
TRELLIS.2 candidates per view	–	1
Object registration		
Translation / yaw / dimension bounds of $\mathcal{B}_j$	–	$\pm 0.18 \max_k e_k^{(0)}$ ± 0.45 rad $[0.75, 1.25]\mathbf{e}^{(0)}$
Surface samples	N_{surf}	70,000 (ICP) 60,000 (silhouette)
Soft-L_1 scale / prior weight	δ, w_{prior}	$0.025/0.03$
ICP tolerance / max. iterations	$\epsilon_{\text{icp}}, N_{\text{icp}}$	$10^{-5}/25$
Function evaluations per inner solve	–	50
Mask resolution factor	–	0.25
Depth tolerance	τ_{depth}	10^{-5}
Empty-boundary penalty	E_{empty}	1
Silhouette weights	$\lambda_{\text{edge}}, \lambda_{\text{worst}}, \lambda_{\text{pc}}$	$0.4/0.35/0.12$
Point-distance scale / cap	$d_{\text{pc}}, \tau_{\text{cap}}$	$0.05/2$
Upper-dimension relaxation	β_{dim}	0.05
Initial translation / dimension / yaw steps	–	$0.035 \max_k e_k^{(0)}$ $0.045\mathbf{e}^{(0)}$ 0.045 rad
Search levels / sweeps per level	$N_{\text{lvl}}, N_{\text{sweep}}$	$3/4$
Step schedule across levels	–	$1, 0.5, 0.25$
Acceptance tolerance	ϵ_{cs}	10^{-5}
Agent		
Max. revision rounds per stage	–	3

Construction. The agent inspects the video, selects informative views, and invokes the modeling and generation tools: it initializes the shell and chooses the parameters to optimize, prepares generation inputs, compares candidate assets in the scene, and adjusts their placement.

Verification. The agent compares source frames with textured and textureless renderings from the same cameras, together with overlays of extra and missing regions and measurements of mask overlap, boundary distance, and point-to-surface distance. It checks structural completeness, object identity, proportions, orientation, and support and occlusion relations. Rigid major structures must agree at their boundaries, whereas soft or intricate components are judged by overall volume, color, and arrangement. Only the original frames serve as the reference; edited completions are used solely to generate candidates. Each discrepancy is localized to one component and mapped to an operation: constrained transforms for pose and scale errors, part-wise deformation for local shape errors, surface refitting for shell mismatches, geometry correction or candidate replacement for structural errors, and material adjustment for appearance errors.

Orchestration. Each revision yields a candidate scene that is evaluated from the relevant source views and accepted only if it improves the diagnosed discrepancy without regressions in other views; a lower aggregate error cannot override a missing structure, severe interpenetration, or incorrect support. Otherwise, the last accepted state is restored. The shell is reviewed before any furniture is built, and a change to a supporting object triggers new checks of everything it supports. When the evidence is ambiguous, the agent searches the video for clearer or complementary views before editing further.



Figure 7: Full qualitative multiview comparison for the four scenes of Figure 6. Each scene is shown from three input-camera viewpoints. The first five columns share camera poses and intrinsics within each row; Ours NVS shows Blender scene overviews.

B Additional Qualitative Results

Figures 8–11 show reconstructions of 23 scenes: seven from our scene collection, eight from Replica, and eight from Imaginarium.

B.1 Our Scene Collection



Figure 8: **Our scene collection, Scenes 1–4.** Each row shows an input frame followed by five renderings of the reconstructed scene.



Figure 9: **Our scene collection, Scenes 13–15.** Layout as in Figure 8.

B.2 REPLICA SCENES



Figure 10: **Replica scenes.** Layout as in Figure 8.

B.3 IMAGINARIUM SCENES

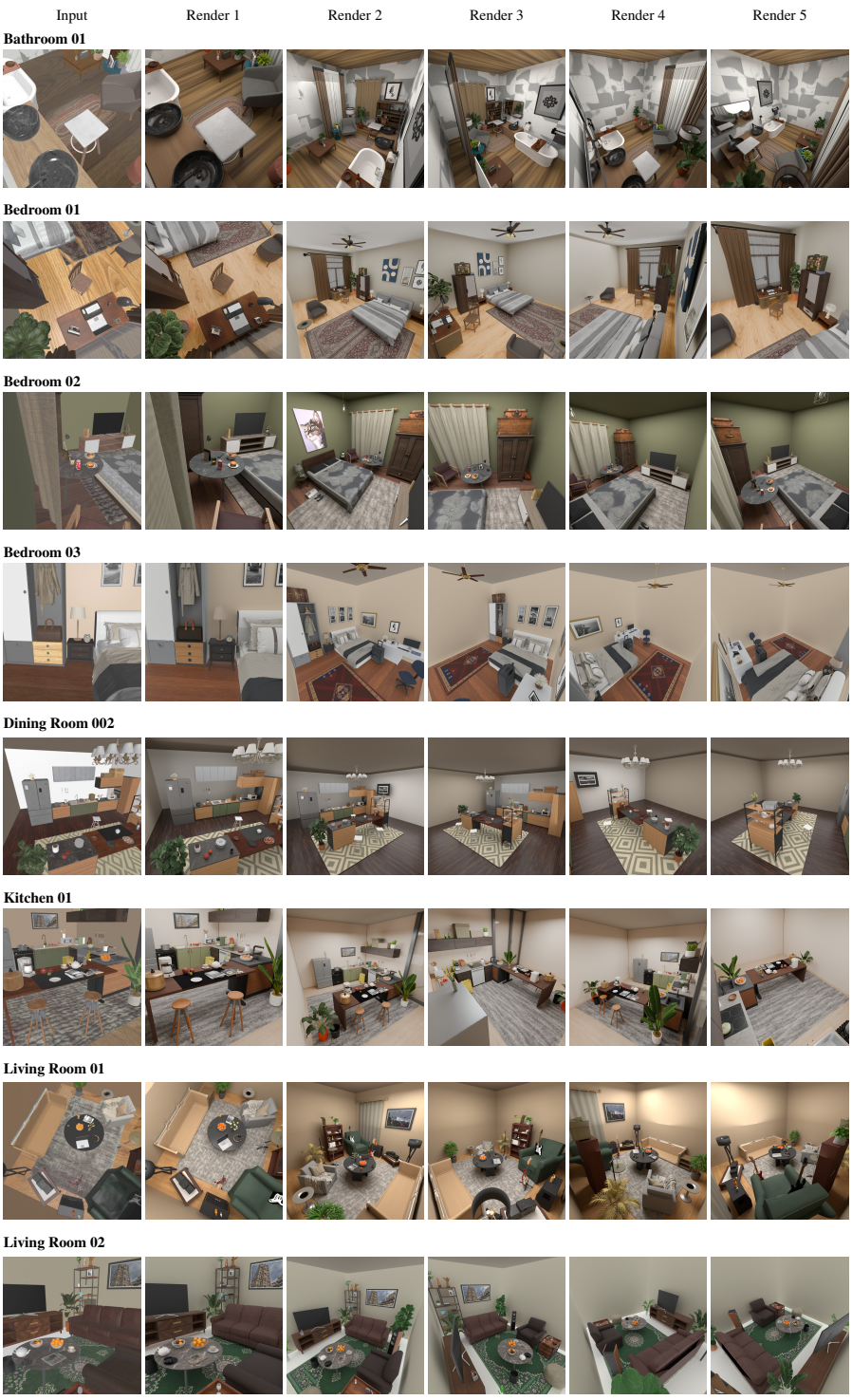

Figure 11: **Imaginarium scenes results.**