ShellMaker: Language-Guided Exterior Completion under Structural Constraints

Ruiqi Xu[1] and Daniel Aliaga[1]

Purdue University, Lafayette IN 47907, USA
{xu2201,aliaga}@purdue.edu

Abstract. Despite advances in indoor scene generation, synthesizing coherent building exteriors consistent with generated interiors remains largely unexplored. Existing methods can generate floor plans and wall layouts but typically stop at a structural shell, lacking stylistically consistent facades and roofs. Completing these exteriors is challenging because the footprint, wall geometry, and opening semantics must remain fixed—constraints that unconstrained generative models often violate. We introduce ShellMaker, a language-guided exterior completion framework that operates under these structural constraints. Given a building scaffold and a text style prompt, ShellMaker generates a complete exterior mesh with PBR materials by combining parametric roof generation, LLM-based part-aware prompt refinement, joint wall–roof material retrieval, and geometry-aware assembly. Operating on a format-agnostic scaffold representation, ShellMaker generalizes to indoor generators, CityGML, and CAD inputs, while maintaining structural consistency and improving architectural coherence over retrieval and unconstrained generative baselines. The project page is available at `https://ruiqixu37.github.io/ShellMaker_web/`.

Keywords: Scene generation, Urban modeling, LLMs, Procedural Modeling

1 Introduction

Recent indoor scene generation systems have made remarkable progress in automated building content creation. Modern pipelines can synthesize plausible interiors with structured floor plans, consistent wall massing, and explicit door and window openings [6, 16, 21, 26, 27]. However, these systems stop at the building shell, leaving facades, roofs, and stylistic exterior details unspecified. The resulting buildings are therefore incomplete and unsuitable for downstream applications such as digital twins, architectural visualization, or urban modeling.

Exterior completion requires generated assets to strictly follow the fixed footprint and opening semantics, as they will otherwise not align with the interior layout. Although recent advances in controllable 3D generation have improved conditional synthesis [4, 5, 8, 10], these models cannot be reliably applied to exterior completion. This limitation stems from building data scarcity, as large-scale

Fig. 1: ShellMaker. Given a building scaffold and a style text prompt, our system produces a complete, textured 3D exterior. The wall geometry and window/door opening placements are preserved exactly across all style variations.

paired datasets linking structured building layouts to fully textured exterior meshes are not currently available. Consequently, a gap remains between perceptually realistic generation and layout-faithful exterior synthesis.

We address this problem with ShellMaker, a language-guided framework for exterior completion under hard structural constraints. Given a style prompt and a building scaffold specifying wall massing and opening layouts, ShellMaker generates a stylistically coherent exterior while strictly preserving the building footprint and layout semantics with three coordinated stages. First, a structural stage infers the fixed footprint from the scaffold and procedurally generates parametric roof geometry subject to the footprint constraint. Second, a style stage retrieves compatibility-aware wall–roof material pairs and uses a large language model (LLM) to translate the user prompt into part-aware specifications, which are used by pretrained 3D generative models to synthesize detailed, style-consistent architectural elements at the part level. Third, an assembly stage performs geometry-aware opening carving, semantics-aligned asset placement, and scale-consistent UV parameterization, ensuring precise alignment with predefined openings and coherent exterior composition.

Although motivated by indoor scene generators, ShellMaker operates on a format-agnostic scaffold abstraction. This enables, in addition, direct application to CityGML building shells and CAD-derived floor plans without modifying the pipeline. We define quantitative structural consistency metrics that measure footprint violations and layout alignment. Altogether and across diverse scaffolds and style prompts, ShellMaker improves architectural coherence, material harmony, and constraint satisfaction as compared to independent retrieval pipelines and unconstrained generative baselines. Our results and analysis include buildings based on over 200 diverse scaffolds, 500+ textures, and 60 style prompts. Further, ShellMaker integrates seamlessly with indoor scene genera-

tion pipelines, enabling the automatic synthesis of complete building assets with coherent interiors and structurally consistent exteriors.
To summarize ShellMaker's contributions:

1. We formulate the task of language-guided exterior completion under hard scaffold constraints, turning partial building scaffolds into complete, style-consistent assets.
2. We propose a modular pipeline to integrate procedural structure generation, part-level stylization, and compatibility-aware wall–roof material retrieval that limits cross-stage error propagation and enables asset reuse.
3. We introduce a geometry-aware assembly process that enforces footprint, wall-boundary, and opening preservation, yielding editable, part-structured exterior meshes.

2 Related Works

3D Generative Models. Recent advances in text-to-3D and image-to-3D generation have enabled the synthesis of detailed 3D assets from natural language or visual inputs [3,11,22,24,29]. Feed-forward 3D diffusion models and reconstruction pipelines can produce textured meshes with strong appearance fidelity, with some methods additionally predicting physically-based-rendering (PBR) material parameters such as diffuse, normal, and roughness maps [9,17,23]. However, these methods are typically designed for unconstrained generation and do not enforce predefined geometric layouts. As a result, generated geometry often deviates from required structural constraints when applied to structured architectural inputs.

Controllable 3D Generation. Several works introduce controllability through spatial conditions such as depth and normal maps [15, 25], part-level semantics [12, 28], or geometric priors [4, 5, 10]. These approaches improve alignment between generated geometry and input signals, but the control remains approximate and does not guarantee strict preservation of topology or layout semantics. In particular, existing methods lack mechanisms to jointly enforce footprint consistency while maintaining semantic alignment with text prompts. In contrast, our method operates under hard structural constraints, treating the building footprint and opening placements as immutable while generating stylistically consistent exterior components.

Indoor Scene and Building Generation. Recent indoor scene generation systems can synthesize structured interiors with semantically grounded floor plans, wall layouts, and explicit door and window openings [1,6,7,16,18,21,26,27]. However, these pipelines typically stop at the structural shell and do not produce coherent building exteriors. Recent works attempt exterior generation conditioned on coarse geometric priors such as 3D shape hints or semantic maps [2, 8, 10], but they do not fully preserve the input footprint, lack mechanisms to jointly enforce structural constraints and semantic style, and are not designed to operate

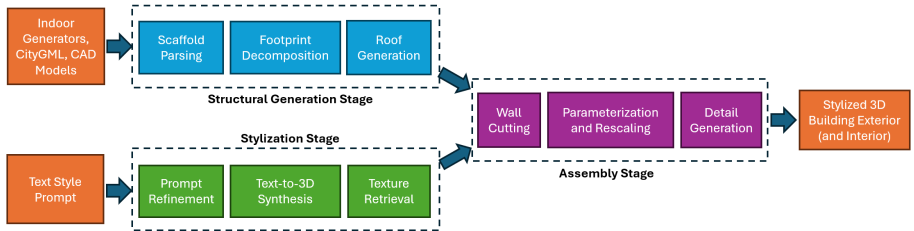

Fig. 2: ShellMaker Pipeline. Given a structured building scaffold (e.g., indoor scene generators, CityGML, or CAD models) and a text style prompt, ShellMaker synthesizes a complete stylized 3D exterior through three stages. The structural stage parses the scaffold and generates parametric roof geometry, while the style stage refines the prompt using an LLM, retrieves compatible wall–roof materials, and generates style-conditioned architectural parts. These components are combined through structure-aware assembly stage, producing a layout-preserving, style-consistent building exterior.

directly on structured outputs from indoor scene generators. Earlier work in procedural urban modeling generates buildings from rule-based grammars but requires manual specification and lacks language-driven style control [13, 14, 20]. In contrast, ShellMaker operates directly on structured building scaffolds produced by indoor generators, CAD models, or CityGML data, enabling the automatic completion of layout-consistent exteriors and the generation of complete building assets.

3 Method

3.1 Problem Formulation

Given a structured building scaffold $\mathcal{S}$ and a text style prompt $\mathcal{P}$, our goal is to generate a textured 3D exterior mesh

$$\mathcal{E} = \Phi(\mathcal{S}, \mathcal{P}), \tag{1}$$

that is both geometrically faithful to the scaffold and semantically aligned with the requested architectural style. The scaffold $\mathcal{S}$ encodes the building footprint, wall boundaries, and the spatial placement of door and window openings, and may be sourced from structured scene graphs produced by deep-learning based indoor scene generators, CAD models, or CityGML files. The output $\mathcal{E}$ is a fully assembled exterior mesh with PBR-ready material parameters, consisting of textured wall surfaces, procedurally generated roof geometry, style-consistent door and window assets inserted into predefined openings, and optional roof ornaments.

To ensure compatibility with indoor scene generators and structured building pipelines, the synthesized exterior must satisfy strict structural consistency. In particular, the footprint topology, wall boundaries, and opening placements specified by the scaffold need to be preserved exactly. Beyond geometry, the exterior should also reflect the semantic style specified by the user prompt.

As illustrated in (Fig. 2), ShellMaker decomposes the synthesis process into three coordinated stages:

$$\Phi = \Phi_{\text{assemble}} \circ \big(\Phi_{\text{structure}},\ \Phi_{\text{style}}\big), \tag{2}$$

where $\Phi_{\text{structure}}(\mathcal{S})$ constructs the geometric scaffold and generates parametric roof geometry from the fixed footprint (Sec. 3.2), $\Phi_{\text{style}}(\mathcal{P})$ translates the user prompt into part-aware specifications using a large language model, synthesizes style-consistent door and window assets with a pretrained 3D generative model, and retrieves compatible wall–roof material pairs (Sec. 3.3), and Φ_{assemble} combines the two stages to produce the final exterior mesh through geometry-aware opening carving, semantics-aligned asset placement, and scale-consistent UV parameterization (Sec. 3.4).

3.2 Structure Generation Stage

Scaffold Parsing. The first stage converts heterogeneous architectural inputs into a unified *scaffold* representation that encodes walls, floors, ceilings, and door/window openings in a common schema. We support three classes of inputs:

1. *Indoor Scene Generators.* The parser extracts wall segments, floor polygons, and door/window opening metadata from the scene graph.
2. *CityGML Models.* The system parses semantically tagged surfaces (e.g., `WallSurface`, `RoofSurface`, `GroundSurface`) and merges coplanar wall fragments that belong to the same façade plane. Openings are projected into each wall's local frame to recover their width, height, and placement.
3. *BIM/CAD Data.* Geometric primitives are tessellated into triangle meshes using `ifcopenshell`. Wall footprints and normals are inferred from mesh geometry, and opening metadata is recovered either from explicit Industry Foundation Classes (IFC) attributes or from mesh-based bounding boxes.

All parsers produce a normalized scaffold representation containing wall segments with 2D endpoints and outward normals, floor and ceiling surfaces, opening placements in both world and wall-local coordinates, and floor-plan polygons for downstream roof generation. This unified scaffold enables the subsequent structural, style, and assembly stages to operate regardless of the input format.

Footprint Decomposition. To generate roofs for complex footprints, we first partition the building footprint into simpler subregions. ShellMaker supports two user-controlled decomposition strategies. The first strategy is straight-skeleton decomposition, which computes the medial axis of the floor-plan polygon and partitions the footprint into cycle regions associated with skeleton ridges. Adjacent regions may be merged under a convexity tolerance (ϵ=0.02) to reduce over-fragmentation. This representation preserves the geometric structure of the

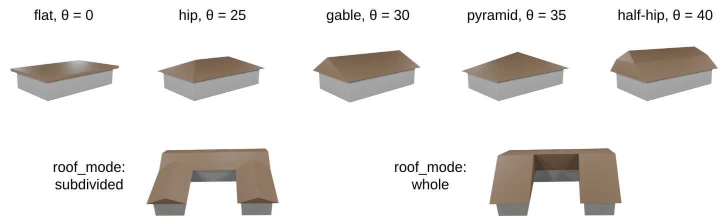

Fig. 3: Parametric Roof Generation. Top: canonical roof types supported by ShellMaker (flat, hip, gable, pyramid, and half-hip) with example pitch angles θ. Bottom: two roof generation modes for complex footprints.

original footprint and naturally supports conformal roof generation for non-convex layouts.

The second strategy enables a wider range of roof compositions by providing an alternative greedy rectangle decomposition that recursively extracts the largest axis-aligned rectangle contained in the footprint, subtracts it, and repeats on the residual polygons until the remaining area falls below a threshold. The resulting rectangular subregions are then assigned independent roof primitives, enabling diverse multi-part roof structures over complex footprints.

Roof Generation. Given a footprint or set of subregions, ShellMaker generates parametric roof geometry from five canonical roof types: *flat*, *gable*, *pyramid*, *hip*, and *half-hip*. Roof geometry is additionally controlled through pitch angle (θ_{pitch}), eave overhang (d_{overhang}), ridge-length fraction (f_{ridge}), and half-hip clipping fraction (f_{hip}), enabling continuous variation in roof profiles.

Like the decomposition strategy, the roof construction mode is a user-selectable option. ShellMaker supports two roof construction modes. In *whole* mode, a single roof is generated over the entire footprint. In *subdivided* mode, roofs are generated independently for each footprint subregion obtained from the decomposition stage and merged into a single mesh, enabling multi-part roof compositions over complex layouts.

3.3 Stylization Stage

Part-aware Prompt Refinement with LLM. As ShellMaker is designed to support free-form text style prompts, inputs such as "a Victorian bakery" describe the building only at a coarse semantic level and provide insufficient guidance for generating detailed architectural parts. Such descriptions rarely specify the geometry or decorative elements of individual components, leading to stylistically inconsistent results when used directly as text-to-3D prompts. Generating high-quality doors, windows, and roof ornaments therefore requires more explicit part-aware prompts that capture stylistic cues at the element level.

To bridge this gap, we employ an instruction-tuned LLM to automatically refine the input free-form prompt into a set of structured text-to-3D prompts.

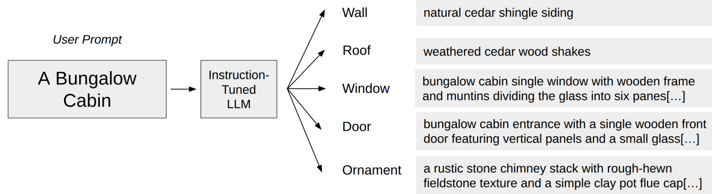

Fig. 4: Part-aware Prompt Refinement. A free-form user prompt is passed to an instruction-tuned LLM, which generates a structured, style-consistent text prompt for each architectural component.

The system instruction enforces that each prompt describes a single isolated architectural element with no surrounding walls or environmental context, ensuring that the generated meshes can be composited cleanly onto the scaffold. The LLM is further guided to include style-authentic geometric and decorative cues appropriate to the architectural style while respecting size constraints for each variant. This refinement process is applied independently to the wall, roof, window, door, and roof-ornament components.

Two-stage Text-to-3D Synthesis. Each refined prompt results in a textured mesh by means of a two-stage pipeline: Nano Banana [19] generates a reference image from the prompt, and Trellis-2 [23] reconstructs a textured mesh with PBR material parameters from the image. This factorization leverages the strong stylistic fidelity of modern image generators while delegating geometric reconstruction to a dedicated 3D backbone, enabling style-consistent asset synthesis without per-style supervision or fine-tuning.

Joint Wall-Roof Texture Retrieval. Rather than generating surface materials from scratch, we adopt a retrieval-based approach for wall and roof textures. Current generative texture models often struggle to produce tileable, physically consistent PBR materials with aligned diffuse, normal, and roughness maps – properties essential for large architectural surfaces. Unfortunately, naive per-surface retrieval can yield perceptually inconsistent combinations, because facade and roof materials exhibit strong cross-surface dependencies in real-world architecture. To address this issue, we introduce a compatibility-aware joint retrieval mechanism that precomputes cross-surface compatibility offline and incorporates it during runtime texture selection.

Offline Compatibility Estimation. Let $\mathcal{W}=\{w_i\}_{i=1}^{N_w}$ and $\mathcal{R}=\{r_j\}_{j=1}^{N_r}$ be PBR texture-pack libraries for walls and roofs (each containing diffuse, normal, and roughness maps). We precompute a compatibility matrix $C \in [0,1]^{N_w \times N_r}$ whose entries combine four signals:

$$C_{ij} = \lambda_1 C_{ij}^{\text{mat}} + \lambda_2 C_{ij}^{\text{color}} + \lambda_3 C_{ij}^{\text{freq}} + \lambda_4 C_{ij}^{\text{clip}}, \qquad \lambda = (0.45,\, 0.25,\, 0.15,\, 0.15). \tag{3}$$

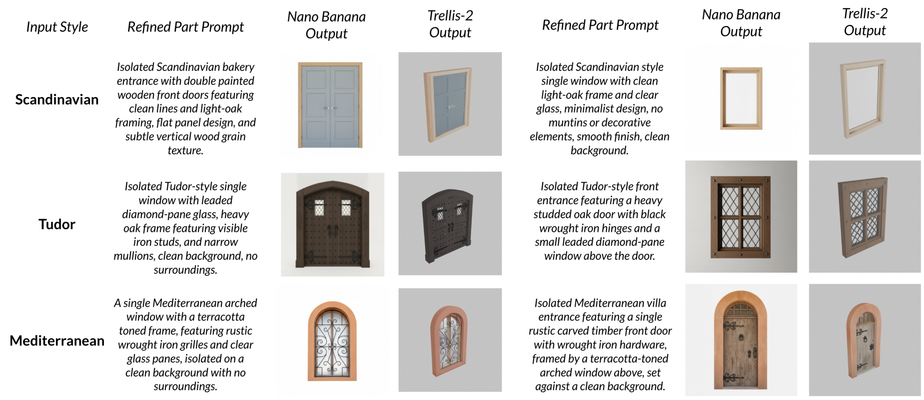

Fig. 5: Two-stage part synthesis intermediates. LLM-refined part prompts, Nano Banana reference images, and reconstructed Trellis-2 meshes across three styles and two element types. Style-specific geometric and decorative cues introduced during prompt refinement carry through to the final textured parts.

C^{mat} assigns high scores to commonly co-occurring architectural pairings (e.g. brick walls with clay-tile roofs, concrete facades with metal roofs). C^{color} measures color harmony via mean $L^*a^*b^*$ distance and saturation difference of the diffuse maps. C^{freq} compares radial power spectra (2D FFT) by cosine similarity, capturing texture-scale alignment. Finally, C^{clip} computes CLIP embedding similarity between diffuse maps as a high-level semantic signal. These scores are computed once offline for all texture pairs.

Runtime Joint Retrieval. At runtime, Shellmaker computes the CLIP similarities of the refined wall and roof prompts t_w and t_r against the texture libraries:

$$s_w(i) = \cos(CLIP(t_w), CLIP(w_i)), s_r(j) = \cos(CLIP(t_r), CLIP(r_j))$$

We first shortlist the top-K wall textures according to $s_w(i)$ and the top-K roof textures according to $s_r(j)$, producing K^2 candidate pairs. Each pair is scored using the precomputed compatibility matrix:

$$s_{ij} = \alpha\, s_w(i) + \beta\, s_r(j) + \gamma\, C_{ij}, \qquad (\alpha, \beta, \gamma) = (0.40,\, 0.30,\, 0.30). \tag{4}$$

To avoid deterministic selections and encourage stylistic variation, we retain the top-M pairs and sample the final pair using a temperature-controlled softmax $p_{ij} \propto \exp(s_{ij}/\tau)$. Lower τ favors conservative combinations while higher τ encourages diversity.

3.4 Assembly Stage

The final stage assembles the scaffold, roof geometry, generated assets, textures, and ornaments into a single mesh, resolving three geometry-aware sub-problems:

UV parameterization, opening cutting with asset insertion, and optional architectural detailing.

Shape-Conforming Wall Cutting. For each inserted part, the pipeline computes a 2D boundary polygon that drives the wall opening cut. It first determines the part's body bounds by trimming the outermost 5% of surface area along each axis, excluding ornamental protrusions (e.g., pediments, pilasters, platforms) that inflate the bounding box. The mesh is then clipped at its back-plane to remove interior-protruding geometry. If the resulting silhouette is sufficiently rectangular (i.e., projected area $\geq$ 95% of its bounding box), a fast rectangular cut is used directly. Otherwise, the mesh is scaled to match the target opening dimensions and a 2D boundary polygon is extracted via a binned silhouette envelope. The X-span is divided into 64 bins and the upper and lower Y-extents per bin are traced into a closed polygon, simplified, and outward-buffered by 0.01 to ensure full coverage. The polygon is finally rescaled to fit exactly within the target opening dimensions before being used as the wall cutter through boolean subtraction.

UV Parametrization and Auto-rescaling. Each wall and roof face is assigned world-space planar UVs by projecting onto the plane perpendicular to the dominant component of its unit normal. Roof meshes use a dedicated *skirt mode*, where horizontal and sloped roof faces use a ground-plane projection, while vertical fascia faces anchor their UVs to the same horizontal coordinates with a vertical offset measured from the roof edge. This prevents visible seams along the eave-slope boundary.

To maintain consistent perceptual texture density across different materials, we apply an automatic UV rescaling factor precomputed offline for each texture. For every diffuse texture map, we estimate its dominant spatial frequency by computing a 2D FFT of the luminance image and performing radial averaging of the power spectrum to obtain a 1D frequency profile. The power-weighted mean frequency (excluding very low-frequency components) characterizes the typical feature scale of the material. Across a texture library, the median dominant frequency is used as a reference; each texture receives a scale factor $s = \text{median}/f_{\text{dominant}}$. Coarse materials (e.g., large brick patterns) therefore repeat more frequently across a surface, producing smaller visible tiles, while fine-grained materials repeat less frequently. At runtime this factor multiplies user-specified UV scale to produce the final tiling frequency used in planar UV assignment, ensuring consistent perceptual material scale across facades and roofs.

Optional Architectural Details. ShellMaker optionally augments the base structure with procedural roof utilities and decorative ornaments. Downspouts are generated through a gutter-to-downspout pipeline that first extracts eave edges from the roof mesh and chains them into continuous gutter polylines. In addition, roof ornaments are generated from a small set of architectural categories

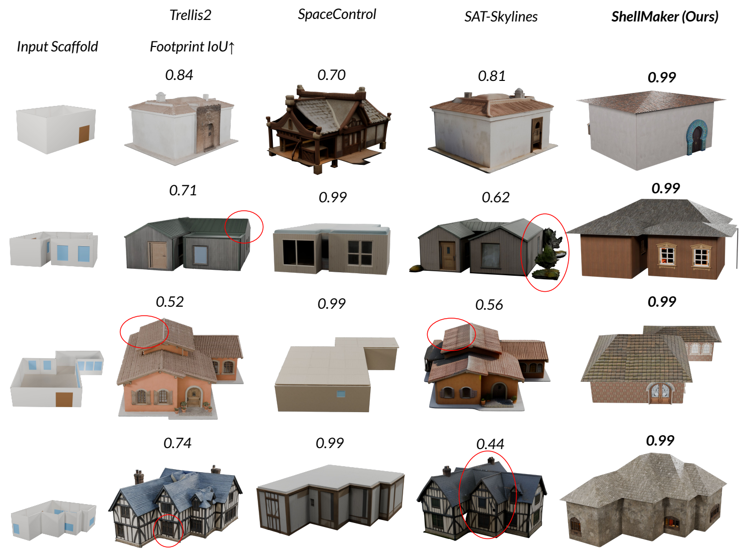

Fig. 6: Given the same scaffold and style prompt, generative baselines often produce visually plausible exteriors but fail to faithfully preserve the scaffold footprint, exhibiting incorrect aspect ratios, omitted footprint boundaries, or geometry extending beyond the scaffold. Methods that better respect the footprint frequently do so at the cost of degraded texture quality. In contrast, ShellMaker preserves the scaffold geometry while producing stylistically consistent exteriors. Red circles highlight subtle footprint mismatches, objects outside the scaffold, and incorrect structural semantics.

(e.g., chimneys, finials, and dormers). For each placement, the ornament is embedded into the roof surface, and clipped against the roof geometry so that only geometry above the roof remains. This clipping step ensures ornaments conform to sloped roofs while preventing intersections with underlying roof panels.

4 Experiments

4.1 Datasets

Building Data. Our primary evaluation uses 100 scaffolds exported from indoor scene generation pipelines, spanning 10 building types (e.g., apartment, bookstore, bakery, gym, ...) with diverse footprint geometries and opening configurations. To test format generality, we additionally source 50 CityGML-derived building shells and 50 CAD-derived models from the Internet, for a total of 200 test scaffolds. All sources are converted into our unified representation.

Wall and Roof Textures. We construct an asset pool of 498 wall textures and 81 roof textures by manually filtering assets from CC0 Textures and Poly Haven. All textures are 4k resolution, include diffuse, normal, and roughness maps, and are released with a free license, enabling PBR of generated buildings.

Table 1: Main quantitative evaluation on style alignment and structural consistency.

	Style Alignment		Structural Consistency		
Method	CLIP-Sim↑	UNI3D↑	Footprt IoU↑	Opening IoU↑	Center L2↓
Trellis-2	**0.318**	0.401	0.492	0.431	0.914
SpaceControl	0.274	0.258	0.812	0.508	0.523
SAT-Skylines	0.281	0.265	0.761	0.548	0.463
ShellMaker	0.315	**0.424**	**0.992**	0.883	0.091
- Joint Texture	0.285	0.357	0.992	0.889	**0.083**
- LLM Prompt	0.293	0.364	0.992	**0.922**	0.095
- Both	0.280	0.305	0.992	0.881	0.084

Prompt Suite. We define 60 prompt templates across three categories: *style-only* (20 templates specifying architectural character without material constraints; e.g., *"a Victorian bakery"*), *material-only* (20 templates specifying wall and roof finishes without a style descriptor; e.g., *"a building with red brick walls and a grey slate roof"*), and *mixed* (20 templates combining a style descriptor with explicit material clauses; e.g., *"Tudor style with red brick walls"*). The full suite is provided in the supplementary material.

4.2 Implementation Details

Prompt refinement and ornament planning use GPT-4.1 with structured JSON output. Reference images are generated with Nano Banana and lifted to textured meshes by Trellis-2. Generated assets are cached per unique <style, category, part-name> tuple, so openings sharing the same canonical part name in the scaffold metadata reuse a single mesh for regularity. For evaluation, all methods are rendered under identical conditions, with 8 azimuth views at 30° elevation in a turntable configuration plus 2 street-level perspective views aligned to principal facades, lit by a fixed neutral High Dynamic Range Imaging (HDRI) environment map with constant exposure. All code and data will be released.

4.3 Main Quantitative Results

We compare ShellMaker with recent 3D generative baselines including Trellis-2 [23], SpaceControl [5], and SAT-Skylines [10]. To provide strong image-based baselines for Trellis-2 and SAT-Skylines, we attempt to align generation with the input scaffold using image-level guidance. We render depth and semantic maps from the scaffold, and prompt Nano Banana to generate a plausible building image conditioned on these signals. The resulting image is then used as the input to Trellis-2 and SAT-Skylines. SAT-Skylines, along with SpaceControl, are methods that extend the Trellis-1 framework by injecting geometric shape priors into the pretrained latent space to encourage consistency with the input structure during generation.

Table 1 reports both style alignment and structural consistency metrics across all methods. Style alignment is measured using CLIP-Sim, computed as the average multi-view CLIP similarity between rendered images and the input prompt, and UNI3D, which provides complementary view-independent evaluation. Structural fidelity is evaluated using footprint IoU, opening alignment IoU, and the average L2 distance between predicted and target opening centers. Opening alignment IoU measures overlap between predicted and scaffold opening regions, capturing agreement in size and extent, but it can be overly strict when the generator produces non-rectilinear openings (e.g., arched windows) that naturally deviate from the scaffold's rectangular semantic box, without being structurally incorrect. We therefore also report the average L2 distance between predicted and target opening centers, which measures positional accuracy independent of shape/area.

Generative baselines achieve reasonable prompt alignment but often fail to satisfy structural constraints. Trellis-2 achieves the highest CLIP-Sim among all methods (0.318) but scores poorly on structural metrics, with a footprint IoU of only 0.492 and an opening center error of 0.914, confirming that image-conditioned 3D reconstruction cannot enforce hard geometric constraints. SpaceControl and SAT-Skylines improve structural fidelity by injecting geometric priors into the latent space, yet both still exhibit substantial opening misalignment and suffer notable drops in style alignment relative to Trellis-2.

In contrast, ShellMaker achieves strong performance across both style and structural metrics. It attains a footprint IoU of 0.992 and an opening center error of only 0.091, indicating near-perfect preservation of scaffold boundaries and opening placement, which makes the method well suited for integration with indoor scene pipelines. ShellMaker produces stylized architectural elements (e.g., arched windows or decorative doors), which may deviate from the rectangular semantic boxes provided by the scaffold; consequently, the default prompt ablation, which always generate simpler rectangular openings, can achieve slightly higher Opening IoU. However, when considered together with the center error metric, the results show that ShellMaker still maintains highly accurate opening placement. On style alignment, ShellMaker matches Trellis-2's CLIP-Sim while substantially outperforming all baselines on the view-independent UNI3D signal. Removing joint texture selection reduces CLIP-Sim from 0.315 to 0.285 and UNI3D from 0.424 to 0.357, while disabling LLM-based prompt refinement similarly degrades both style scores, with the largest drop observed when both are removed. Structural consistency remains stable across all ablations, and footprint IoU is identical across ShellMaker variants because the pipeline never alters the underlying wall massing. Roofs and facade elements are generated while the scaffold footprint and wall geometry remain fixed.

4.4 Perceptual Evaluation

We conduct a two-alternative forced choice (2AFC) evaluation using GPT-4.1 as the judge across 100 scaffold–prompt pairs. For each pair, the judge is presented with the input prompt and rendered views of ShellMaker alongside a random

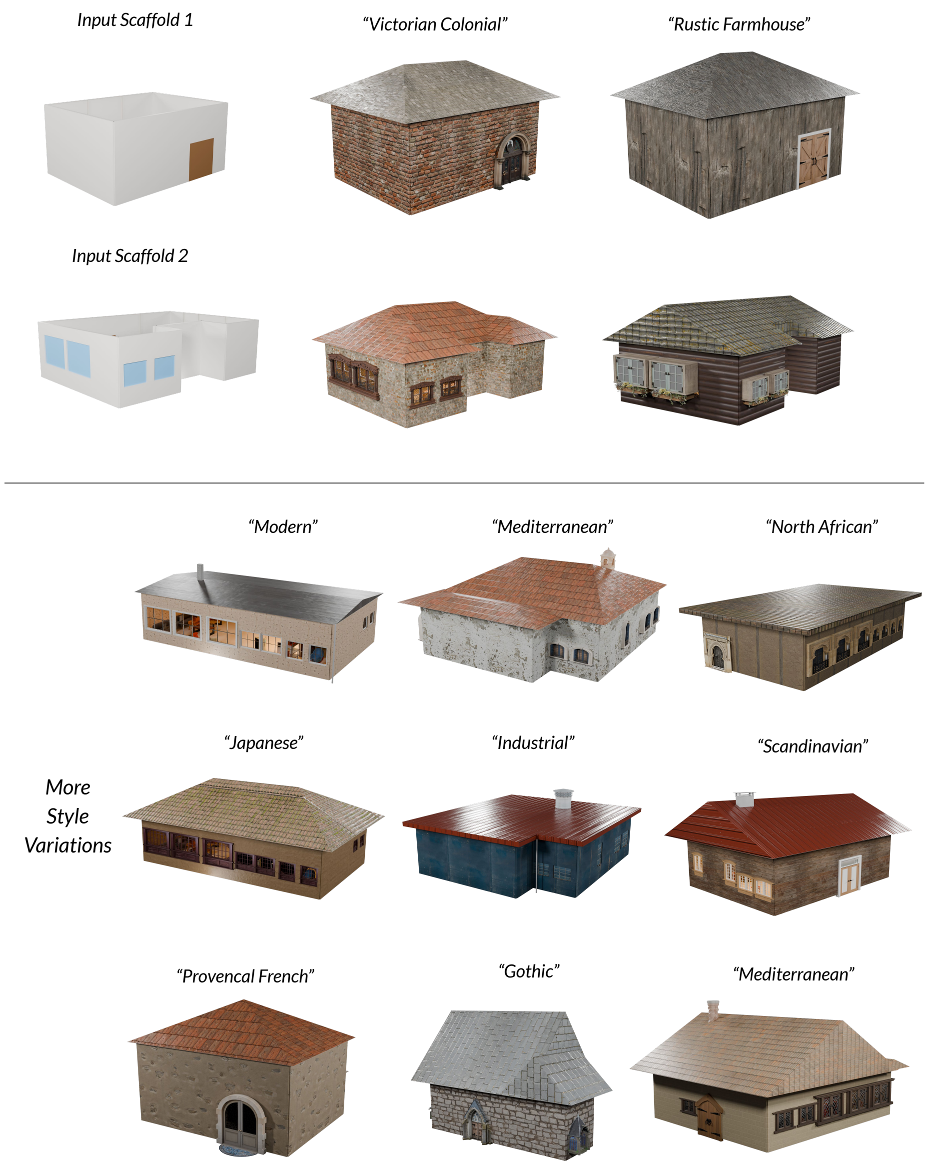

Fig. 7: Style gallery. (Top) Two input scaffolds each rendered under two style prompts, demonstrating preservation of wall geometry and opening placements across variants. (Bottom) Nine additional styles applied to various scaffolds, illustrating the breadth of architectural styles ShellMaker can automatically generate with appropriate roof forms, materials, and detailing.

Table 2: Additional evaluation results.

(a) Generalization across scaffold sources.

	Style		Structural	
Source	CLIP-S↑	UNI3D↑	Ftpr. IoU↑	Open. L2↓
Indoor Generator	0.315	0.424	0.992	0.091
CityGML	0.308	0.418	0.991	0.104
CAD	0.301	0.409	0.889	0.118

(b) LLM evaluation (2AFC, %).

Method	Overall↑	Style↑	Struct.↑
Trellis-2	71.2	48.3	84.1
SpaceControl	80.1	71.2	92.8
SAT-Skylines	87.4	68.8	91.2

baseline, and asked to select the preferred output on overall quality, style fidelity, and structural plausibility.

ShellMaker achieves win rates above 70% overall against all three baselines (Table 2b). Against Trellis-2, gains are most pronounced on structural plausibility (84.1%), while style fidelity is more competitive (48.3%), reflecting Trellis-2's stronger appearance modelling but weaker geometric grounding. Against SpaceControl and SAT-Skylines, ShellMaker leads consistently across all axes, with particularly large margins on structural plausibility (92.8%, 91.2%), confirming that ShellMaker produces outputs that better respect input scaffold constraints.

4.5 Generalization Across Scaffold Sources

Table 2a evaluates ShellMaker on scaffolds derived from different structured building formats. Results on indoor generator inputs closely match the main evaluation, confirming that the system performs consistently on the scaffold representation produced by indoor scene generation pipelines. On CityGML inputs, style alignment decreases slightly while structural metrics remain nearly unchanged. Finally, CAD inputs exhibit noisier geometry and inconsistent topology, leading to a modest drop in footprint IoU and higher opening alignment error. Despite these challenges, ShellMaker maintains competitive style scores and preserves most structural constraints, indicating that the pipeline generalizes well across heterogeneous scaffold sources.

4.6 Qualitative Results

Figure 7 presents representative outputs generated from indoor-scene scaffolds with different style prompts. The top examples show that ShellMaker preserves the input footprint and maintains accurate alignment of door and window openings while adapting roof geometry, materials, and architectural details to the requested style. The bottom gallery further demonstrates the stylistic range of the system across a variety of architectural themes, highlighting the ability of ShellMaker to produce diverse exteriors while respecting structural constraints. We provide more qualitative results and a fly-through video in the supplemental materials.

4.7 Limitations

ShellMaker's primary failure mode lies in the part generation pipeline, which can occasionally produce degenerate components such as malformed roof segments or geometrically inconsistent parts. In addition, part intersections may occur in semantically dense regions of the facade where multiple elements are placed in close proximity. We refer the reader to the supplementary material for illustrative failure cases.

ShellMaker currently treats the facade as a single massing shell and stylizes openings uniformly without modeling per-floor articulation. Richer multi-floor facade variation, such as distinct ground-floor treatments or story-dependent window styles, remains beyond the current scope.

4.8 Future Work

ShellMaker currently assumes structured building scaffolds with predefined wall and opening semantics. It relies on existing layout specifications rather than generating structural layouts directly. Integrating learned 3D layout generation models could enable joint synthesis of exterior geometry and opening configurations within a unified framework. Moreover, in this work, interior–exterior consistency refers to structural preservation (footprint, wall boundaries, and openings) together with prompt-driven stylistic matching of the exterior. Extending this to align generated exteriors with specific interior material and furnishing choices through scaffold refinement is a promising direction for future work.

ShellMaker also retrieves wall and roof textures from curated material libraries to ensure architectural realism and physical plausibility. While effective, this retrieval-based approach limits stylistic diversity. Future work could incorporate generative texture synthesis models to support more controllable, stylized, or unconventional architectural appearances.

5 Conclusions

We introduced ShellMaker, a language-guided framework for completing building exteriors from structured scaffolds while strictly preserving structural constraints. Given a scaffold and a style prompt, the system combines parametric roof generation, LLM-based prompt refinement, compatibility-aware material retrieval, and geometry-aware assembly to synthesize coherent exteriors. Overall, ShellMaker provides a practical step toward automated generation of complete building assets with consistent interiors and structurally valid exteriors.

6 Acknowledgment

This work is partially supported by NSF Grant #2411273 and NSF Grant #2107096.

References

1. Bokhovkin, A., Meng, Q., Tulsiani, S., Dai, A.: Scenefactor: Factored latent 3d diffusion for controllable 3d scene generation (2024)
2. Chang, K.H., Cheng, C.Y., Luo, J., Murata, S., Nourbakhsh, M., Tsuji, Y.: Building-gan: Graph-conditioned architectural volumetric design generation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 11956–11965 (2021)
3. Chen, X., Chu, F.J., Gleize, P., Liang, K.J., Sax, A., Tang, H., Wang, W., Guo, M., Hardin, T., Li, X., et al.: Sam 3d: 3dfy anything in images. arXiv preprint arXiv:2511.16624 (2025)
4. Dong, W., Yang, B., Ma, L., Liu, X., Cui, L., Bao, H., Ma, Y., Cui, Z.: Coin3d: Controllable and interactive 3d assets generation with proxy-guided conditioning. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–10 (2024)
5. Fedele, E., Engelmann, F., Huang, I., Litany, O., Pollefeys, M., Guibas, L.: Spacecontrol: Introducing test-time spatial control to 3d generative modeling. arXiv preprint arXiv:2512.05343 (2025)
6. Gu, Z., Cui, Y., Li, Z., Wei, F., Ge, Y., Gu, J., Liu, M.Y., Davis, A., Ding, Y.: Artiscene: Language-driven artistic 3d scene generation through image intermediary. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 2891–2901 (2025)
7. Gumin, M., Han, D.H., Yoo, S.J., Ganeshan, A., Jones, R.K., Fu, K., Aguina-Kang, R., Morris, S., Ritchie, D.: Procedural scene programs for open-universe scene generation: Llm-free error correction via program search. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers. pp. 1–11 (2025)
8. Huang, J., Wang, C., Li, L., Huang, C., Dai, Q., Xu, W.: Buildingblock: A hybrid approach for structured building generation. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–11 (2025)
9. Hunyuan3D, T., Yang, S., Yang, M., Feng, Y., Huang, X., Zhang, S., He, Z., Luo, D., Liu, H., Zhao, Y., et al.: Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. arXiv preprint arXiv:2506.15442 (2025)
10. Jin, Z., Feng, A.: Sat-skylines: 3d building generation from satellite imagery and coarse geometric priors. arXiv preprint arXiv:2508.18531 (2025)
11. Lin, J., Yang, X., Chen, M., Xu, Y., Yan, D., Wu, L., Xu, X., Xu, L., Zhang, S., Chen, Y.C.: Kiss3dgen: Repurposing image diffusion models for 3d asset generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5870–5880 (2025)
12. Lin, Y., Lin, C., Pan, P., Yan, H., Feng, Y., Mu, Y., Fragkiadaki, K.: Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers (2025), https://arxiv.org/abs/2506.05573
13. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. In: ACM SIGGRAPH 2006 Papers, pp. 614–623 (2006)
14. Nishida, G., Bousseau, A., Aliaga, D.G.: Procedural modeling of a building from a single image. In: Computer Graphics Forum. vol. 37, pp. 415–429. Wiley Online Library (2018)
15. Qiu, L., Chen, G., Gu, X., Zuo, Q., Xu, M., Wu, Y., Yuan, W., Dong, Z., Bo, L., Han, X.: Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9914–9925 (2024)

16. Raistrick, A., Mei, L., Kayan, K., Yan, D., Zuo, Y., Han, B., Wen, H., Parakh, M., Alexandropoulos, S., Lipson, L., Ma, Z., Deng, J.: Infinigen indoors: Photorealistic indoor scenes using procedural generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 21783–21794 (June 2024)
17. Seed, B.: Seed3d 1.0: From images to high-fidelity simulation-ready 3d assets. arXiv (2025)
18. Tam, H.I.I., Pun, H.I.D., Wang, A.T., Chang, A.X., Savva, M.: SceneEval: Evaluating semantic coherence in text-conditioned 3D indoor scene synthesis (2025)
19. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)
20. Vanegas, C.A., Aliaga, D.G., Benes, B.: Building reconstruction using manhattan-world grammars. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 358–365. IEEE (2010)
21. Wang, X., Yeshwanth, C., Nießner, M.: Sceneformer: Indoor scene generation with transformers. arXiv preprint arXiv:2012.09793 (2020)
22. Wu, S., Lin, Y., Zhang, F., Zeng, Y., Yang, Y., Bao, Y., Qian, J., Zhu, S., Cao, X., Torr, P., et al.: Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. arXiv preprint arXiv:2505.17412 (2025)
23. Xiang, J., Chen, X., Xu, S., Wang, R., Lv, Z., Deng, Y., Zhu, H., Dong, Y., Zhao, H., Yuan, N.J., et al.: Native and compact structured latents for 3d generation. arXiv preprint arXiv:2512.14692 (2025)
24. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 21469–21480 (2025)
25. Xie, Y., Xu, C., Rakotosaona, M.J., Rim, P., Tombari, F., Keutzer, K., Tomizuka, M., Zhan, W.: Sparsefusion: Fusing multi-modal sparse representations for multi-sensor 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17591–17602 (2023)
26. Yang, Y., Jia, B., Zhang, S., Huang, S.: Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. In: Advances in Neural Information Processing Systems (NeurIPS) (2025)
27. Yang, Y., Sun, F.Y., Weihs, L., VanderBilt, E., Herrasti, A., Han, W., Wu, J., Haber, N., Krishna, R., Liu, L., et al.: Holodeck: Language guided generation of 3d embodied ai environments. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16227–16237 (2024)
28. Yang, Y., Guo, Y.C., Huang, Y., Zou, Z.X., Yu, Z., Li, Y., Cao, Y.P., Liu, X.: Holopart: Generative 3d part amodal segmentation. arXiv preprint arXiv:2504.07943 (2025)
29. Ye, C., Wu, Y., Lu, Z., Chang, J., Guo, X., Zhou, J., Zhao, H., Han, X.: Hi3dgen: High-fidelity 3d geometry generation from images via normal bridging. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 25050–25061 (2025)