HiLL PUBLISHING

Engineering Advances, 2026, 6(3), 214-218
https://www.hillpublisher.com/journals/ea/
ISSN Online: 2768-7961

Research on Large Language Model-driven Text-to-3D Virtual Scene Generation Methods

Qingyao Jia

Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA.

How to cite this paper: Qingyao Jia. (2026) Research on Large Language Model-driven Text-to-3D Virtual Scene Generation Methods. *Engineering Advances*, 6(3), 214-218.
DOI: 10.26855/ea.2026.09.013

Received: July 9, 2026
Accepted: August 6, 2026
Published: September 2, 2026

***Corresponding author:** Qingyao Jia, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA.

© 2026 by the author(s).
This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives (CC BY-NC-ND) license, which permits non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited and is not modified or adapted.
https://creativecommons.org/licenses/by-nc-nd/4.0/

Abstract

Text-driven 3D virtual scene generation requires coordinated handling of object semantics, spatial relations, scale, geometry, asset selection, and cross-view consistency. A three-layer framework is proposed in which a large language model supports prompt normalization, entity and relation extraction, and scene-graph construction, while downstream modules address constraint translation, asset retrieval or generation, scene instantiation, geometric representation, and multi-view inspection. 3D-FRONT and Objaverse are used as reference data sources, and published Holodeck results provide component-level evidence for fixed-object layout evaluation. The framework is designed to expose intermediate states, preserve relation provenance, and route detected errors to the processing stage responsible for them. Semantic omissions, infeasible placements, asset mismatches, visibility errors, and geometric inconsistencies are treated as separate failure types rather than merged into one score. The proposed architecture remains conceptual and does not claim end-to-end experimental validation. Its correction accuracy, runtime behavior, scalability, and reproducibility require implementation, compatible baselines, ablation studies, generated scene records, and complete execution logs.

Keywords

Large language model; Text-to-3D; virtual scene generation; scene graph; spatial constraint; 3D asset retrieval

Introduction

Virtual scene generation is becoming part of the production workflow for game development, digital filmmaking, immersive training, and embodied AI, where the task has expanded from isolated object creation to complete spaces shaped by structure, function, and interaction. Existing text-to-3D methods cover diffusion priors, neural fields, and explicit mesh representations [1], yet complex prompts still leave quantities, orientations, support relations, circulation areas, and stylistic constraints only loosely tied to 3D coordinates. Large language models are useful for decomposing long descriptions and supplying scene knowledge; their strongest role lies in semantic compilation, relation encoding, and revision control, while geometric accuracy and physical plausibility still depend on constraint solving and multi-view verification.

1. Technical Mechanism of LLM-driven Text-to-3D Virtual Scene Generation

1.1 Semantic Parsing and Scene Structure Mapping

When a free-form prompt enters a 3D generation pipeline, the LLM first identifies the scene category, object entities, quantities, visual attributes, spatial relations, and functional requirements. It also resolves pronouns, omitted elements, and implied objects, turning phrases such as “beside the window,” “facing the desk,” and “inside the cabinet”

into normalized spatial predicates. The parsed output is stored as a scene graph, where objects form nodes and support, containment, proximity, and orientation form directed edges; each node also carries a size range, material tag, placement priority, and asset source, while Text2NeRF links textual content priors with depth-based geometric priors and offers a practical basis for mapping semantic descriptions into continuous 3D space [2].

1.2 3D Representation Selection and Generation Coupling

The geometric form, interaction demand, and viewing distance of each scene element determine its 3D representation. Walls, floors, and stairs are better handled with procedural meshes; common furniture can be retrieved as editable assets with stable topology, and irregular or long-tail objects may rely on implicit fields or generative models. Before assembly, all outputs require aligned units, coordinate axes, object origins, forward directions, and texture formats; EXIM controls topology with an explicit representation and then refines surface geometry and color with an implicit field, which provides a useful route for balancing structural stability with local detail continuity [3].

2. Hierarchical Method for LLM-driven Text-to-3D Virtual Scene Generation

2.1 Prompt Normalization and Scene Graph Construction

In the proposed workflow, a raw prompt would be normalized into seven fields: scene_type, room_boundary, object_list, attribute_set, relation_set, functional_constraint, and global_style. The LLM would resolve references, verify quantities, distinguish explicit from inferred objects, and map expressions such as “next to,” “facing,” and “inside” to a controlled predicate set. Object nodes would retain category, size, material, parent node, placement priority, candidate source, confidence, and provenance, while relation edges would record direction, constraint level, and edit status. The resulting JSON scene graph would link language parsing with downstream processing: incomplete fields would return for repair, conflicting relations would enter rule-based review, and validated content would proceed to constraint translation, following the structured layout representation used in LayoutGPT [4].

As shown in Figure 1, the upper layer covers prompt normalization, entity resolution, relation extraction, and scene-graph encoding; the middle layer handles constraint translation, anchor placement, asset routing, and layout optimization; the lower layer performs geometric representation, multi-view rendering, inspection, and quality scoring. Layout errors would return to relation processing, while geometry errors would return to the relevant asset or representation stage.

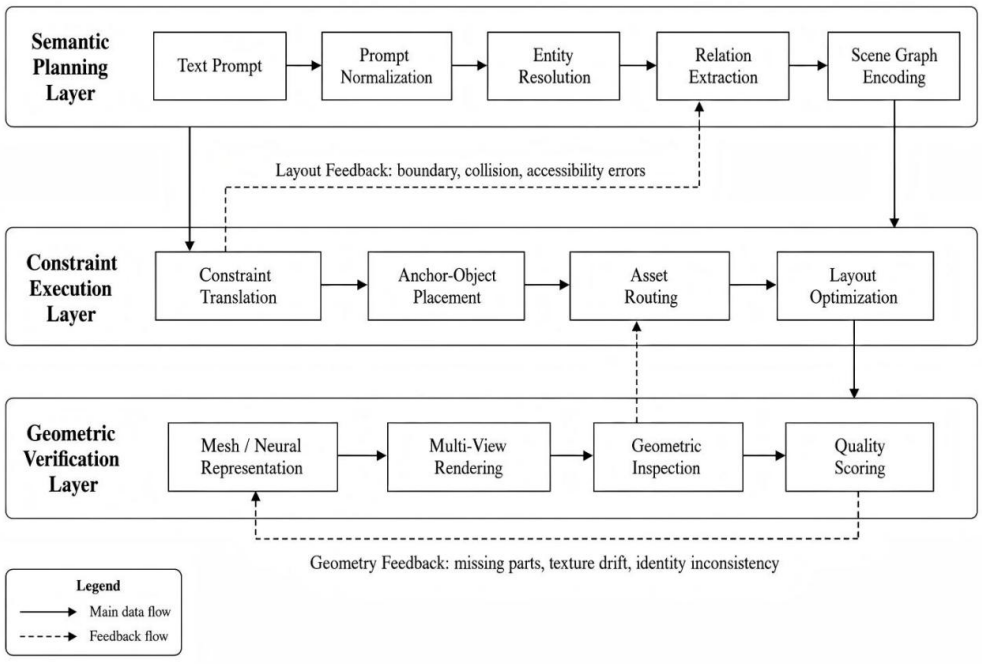

Figure 1. Three-layer dual-feedback architecture for LLM-driven text-to-3D virtual scene generation.

2.2 Spatial Relation Constraints and Object Layout Optimization

Each scene object $o_i$is represented by an oriented bounding stateb_i=$(x_i,y_i,z_i,w_i,d_i,\mathrm{h}_i,\theta_i)$, where (x_i,y_i,z_i) denotes the object center in scene coordinates, (w_i,d_i,h_i) gives its physical dimensions, and θ_i defines yaw orientation. In the proposed solver, boundary compliance, structural support, collision avoidance, stability, and circulation clearance define a feasible set F; these conditions are checked before any preference-based optimization. Broad-phase collision screening would use oriented bounding boxes or spatial indexing, followed by narrow-phase tests with collision proxies, convex hulls, meshes, or signed-distance fields, together with tolerances and intended-contact rules for supported or adjacent objects.

Within the feasible set, layout selection would minimize:

$$\min_{\mathbf{b}\in\mathcal{F}} L_{\text{soft}}(\mathbf{b}) = \lambda_r L_{\text{relation}} + \lambda_p L_{\text{proximity}} + \lambda_o L_{\text{orientation}} + \lambda_v L_{\text{visual}}$$

where the soft terms describe semantic relations, preferred distance, functional orientation, and visual balance. The solver would place anchor objects such as beds, desks, and sofas first, then follow support and adjacency relations to position dependent objects before adding decorative nodes. Safety, stability, boundary, support, collision, and accessibility conditions receive the highest priority, followed by explicit user preferences and LLM-inferred decorative relations; when no feasible layout exists, the solver would relax only inferred soft relations, record the violated conditions, and return the unresolved constraints for revision. Iteration would stop after a feasible layout shows no further objective improvement or reaches a predefined maximum number of updates.

2.3 3D Asset Assembly and View-Feedback Correction

Within the proposed workflow, common objects would be retrieved from public repositories, uncommon objects would be assigned to text-to-3D generation, and regular architectural components would use procedural meshes. Candidates would be screened for file integrity, geometry quality, metadata completeness, and licensing status, then ranked by semantic relevance, dimensional fit, topology, texture availability, and interaction metadata. During scene instantiation, selected assets, solved transforms, materials, collision proxies, and procedural geometry would be combined into an executable scene state and normalized to shared units, axes, origins, orientations, and texture formats. Geometry density would then follow viewing distance, consistent with the mesh-controllable LOD principle in [5].

The instantiated scene state and object registry would serve as the authoritative records for object existence, while a fixed camera path would produce RGB, depth, normal, and object-ID renders for visibility and appearance inspection. An object missing from the scene registry would be separated from one that is present but occluded, outside camera coverage, or absent from a particular rendered view; collision checks would begin with bounding-box or spatial broad-phase filtering and proceed to geometry-aware narrow-phase tests using collision proxies, convex hulls, meshes, or signed-distance fields. Cross-view inspection would rely on stable object IDs, camera-pose projection, visibility reasoning, and geometry- or feature-aware comparison before assigning an error to an asset, material, relation edge, or local surface region. The multi-view priors, camera-pose estimation, Gaussian optimization, and mesh extraction used in 3DS-Gen provide component-level support for this inspection stage [6], while the accuracy of the proposed correction loop still requires direct implementation and experimental evaluation.

3. Published Benchmark Evidence and Proposed Evaluation Protocol

3.1 Public Datasets and Proposed Evaluation Setup

Public verification is provided by 3D-FRONT with information such as room boundaries, object categories, three-dimensional dimensions, spatial positions, rotation angles, and material properties. This dataset includes 6,813 sets of CAD residences, 18,968 rooms with furniture, 31 types of scenes, and 13,151 high-quality texture furniture models [7]; Objaverse 1.0 provides over 800,000 three-dimensional objects with descriptions, labels, or animation information, which are used to expand the range of candidate assets [8]. After loading the data, the categories, length units, model origins, front directions, and bounding box formats need to be standardized, and the room numbers, asset numbers, original transformation parameters, and filtering records should be retained. This enables each generated result to be traced back to the corresponding scene and object.

The comparison scheme is set to include four groups: LLM absolute coordinate output, random collision-free placement, placement along the wall edge, and relationship constraint solving. Each group uses the same room boundaries, object list, candidate assets, and top-down camera, only changing the generation method of object

positions and orientations. The basic prompt only provides the category and quantity; the relationship prompt adds support, proximity, and orientation conditions, and the composite prompt adds the function and style fields. The experimental logs record the parsing results, layout JSON, asset mapping table, and renderings. When the real running files are missing, only the public benchmark is sorted out, and the literature data is not rewritten as self-obtained experimental results.

3.2 Published Benchmark Evidence and Layout Evaluation

Layout quality is evaluated with Mean Reciprocal Rank (MRR). Evaluators inspect four shuffled top-down views generated by different layout strategies and rank them according to boundary violations, object collisions, reachable space, and spatial realism; object categories, object counts, and scene-type agreement are examined separately, while texture fidelity and mesh accuracy remain outside this layout-ranking task. The published results are presented in Table 1.

Table 1. Published MRR results of different object layout strategies

Layout strategy	Bathroom	Bedroom	Kitchen	Living Room	Average
Absolute position	0.369	0.343	0.407	0.336	0.364
Random placement	0.422	0.339	0.367	0.348	0.369
Edge placement	0.596	0.657	0.655	0.672	0.645
Constraint-based	0.696	0.745	0.654	0.728	0.706

Note: The values are taken from the published layout ablation results of Holodeck and are reported benchmarks rather than newly produced experimental results [9].

As shown in Table 1, constraint-based placement records the highest average MRR at 0.706, which is 0.342 higher than the 0.364 obtained by absolute-position prediction, and it ranks first for bathrooms, bedrooms, and living rooms. In kitchens, edge placement reaches 0.655, and constraint-based placement reaches 0.654; the difference of 0.001 is too small to support a meaningful advantage without uncertainty estimates or statistical testing, while MRR remains a relative human-ranking statistic among four layout strategies using the same object inventory. Holodeck reports three user studies involving 680 participants in total; within the residential comparison covering 120 paired scenes, participants preferred Holodeck in 59.8% of asset-selection comparisons, 56.9% of layout-coherence comparisons, and 64.4% of overall comparisons.

3.3 Proposed Failure Detection and Methodological Boundaries

The proposed failure-inspection procedure would read the authoritative scene graph and object registry before examining bounding boxes, support relations, object-ID renderings, depth maps, and multi-view images. A required node absent from the scene state would be labeled missing_from_scene, whereas a registered object absent from a rendered ID view would be classified as present_but_not_visible, unexpected_occlusion, or outside_camera_coverage; collision checking would use bounding-box overlap only as a broad-phase filter, followed by geometry-aware narrow-phase testing with collision proxies, meshes, convex hulls, or signed-distance fields, together with tolerances and intended-contact exceptions. Support detachment would be detected from the gap between the object contact surface and its assigned support surface, while cross-view faults would be assessed with stable object IDs, camera-pose projection, visibility reasoning, and geometry- or feature-aware comparisons; each record would retain node_id, error_type, view_id, relation_id, and revision_target so that correction could be routed to the affected graph node, relation, asset, material, or local geometry region.

The evidence register would classify numerical information as dataset statistics, published benchmarks, or reproduced results and retain the dataset version, prompt, asset pool, model version, camera trajectory, computing device, configuration file, and output path. 3D-FRONT supports controlled evaluation of synthetic indoor layouts, room scales, and furniture combinations, whereas Objaverse supplies open-category assets whose geometry quality, metadata completeness, coordinate normalization, and licensing status require separate checks; neither source represents direct evidence from scanned physical environments. Cross-paper results obtained with different prompts, object inventories, camera settings, rendering conditions, or evaluation protocols should remain in separate comparison groups, and reproduced tables should contain only values supported by executable logs, retained configurations, generated scene files, and verifiable output records.

4. Conclusion

Large language models support semantic decomposition, entity and relation extraction, scene-graph construction, and revision-instruction generation, while geometric solvers, asset libraries, scene-instantiation modules, and renderers handle scale calibration, feasible placement, collision testing, representation assembly, and multi-view inspection. The architecture remains a conceptual framework rather than an experimentally validated generation system; its correction accuracy, editing efficiency, runtime behavior, scalability, and convergence have yet to be established, and the Holodeck results provide component-level evidence only for fixed-object constraint-based layout under the original evaluation protocol. Future work should implement the full data flow, compare compatible baselines under shared prompts and asset pools, and report ablations, generated scenes, failure traces, latency, memory use, human evaluation, physical-function tests, and reproducibility materials.

References

[1] Lee HH, Savva M, Chang AX. Text-to-3D shape generation. Comput Graph Forum. 2024;43(2):e15061.

[2] Zhang J, Li X, Wan Z, et al. Text2NeRF: text-driven 3D scene generation with neural radiance fields. IEEE Trans Vis Comput Graph. 2024;30(12):7749-7762.

[3] Liu Z, Hu J, Hui KH, et al. EXIM: a hybrid explicit-implicit representation for text-guided 3D shape generation. ACM Trans Graph. 2023;42(6):228:1-228:12.

[4] Feng W, Zhu W, Fu TJ, et al. LayoutGPT: compositional visual planning and generation with large language models. In: Advances in Neural Information Processing Systems. 2023;36.

[5] Huang D, Wang N, Huang X, et al. Mesh-controllable multi-level-of-detail text-to-3D generation. Comput Graph. 2024;123:104039.

[6] Gu Z, Gao T, Liu H. Text-to-3D scene generation framework: bridging textual descriptions to high-fidelity 3D scenes. Vis Comput Ind Biomed Art. 2025;8:29.

[7] Fu H, Cai B, Gao L, et al. 3D-FRONT: 3D furnished rooms with layouts and semantics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021:10933-10942.

[8] Deitke M, Schwenk D, Salvador J, et al. Objaverse: a universe of annotated 3D objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023:13142-13153.

[9] Yang Y, Sun FY, Weihs 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. 2024:16227-16237.