IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

ThreePod: A Verification-First System for Prompt- and Image-Driven 3D Modelling and Printing

Sharvil R. Wange[1*], Lokesh A. Chaudhary[1], Jay P. Umbarkar[1], Hemant N. Humane[1], Shailesh S. Dhok[2]

Student, Dept. of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, India[1]

Professor, Dept. of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, India[2]

*Corresponding Author

Abstract: Generative 3D systems can produce a plausible object from a sentence or a single photograph, but plausibility is not the same as coherence, editability, correct scale, printability, or honest provenance. This paper presents ThreePod, a local-first, verification-first pipeline that treats every generative proposal as an untrusted candidate. The text route plans atomic actions, generates editable Three.js blocks, and commits a block only after schema, grounding, syntax, live-browser, and geometric checks. The image route segments a subject, reconstructs a shape, measures visible-side silhouette agreement, separates observed from inferred surface, and exports a part-labelled GLB. The print route converts an approved result into a simulated, layer-locked toolhead path with material and time budgets. We report an offline suite of 766 tests (723 passed, 2 skipped, 41 deselected, 0 failed), a ten-prompt scene benchmark with 401 geometry checks and zero failures, and four research image assets with valid GLB exports and visible-side IoU between 0.6016 and 0.9640. A CPU-only comparison of 3B and 7B local code models shows the expected quality–latency trade-off. The contribution is not a better-looking generator; it is an auditable pipeline that makes generation inspectable, measurable, and print-aware.

Keywords: 3D generation, text-to-3D, image-to-3D, verification, printability, LLM code generation, provenance, digital manufacturing.

I. INTRODUCTION

Generative artificial intelligence has made 3D content creation accessible to users who cannot model by hand. Text-to-3D methods use a 2D image prior to optimise a 3D representation [1], [2]; native 3D models generate implicit shapes or structured latents directly [3], [4], [7], [8]; single-image methods lift a photograph into a novel view or a mesh [5], [6]. Commercial services now return textured assets from short prompts. The visual progress is real.

A plausible render, however, is not a usable asset. A design or manufacturing workflow also asks whether the mesh is watertight, whether the dimensions are plausible, whether the object is standing on the floor or floating, whether the generated code contains a syntax error, whether materials are defined consistently, whether the model can be edited without regeneration, which parts of the surface were seen in the photograph, and whether the result can be sliced and printed within a material budget. A single image can look convincing while failing every one of these checks.

Existing systems generally optimise one part of this problem. Hosted generators optimise visual quality and speed. Research models optimise shape or texture fidelity. LLM code agents such as BlenderGPT and SceneCraft generate editable programs, but they usually trust the generated program once it runs [9], [10]. CAD tools and slicers provide exact geometry and machine paths, but they do not generate a scene or an object from a prompt or a photo [11], [12], [13], [14]. Verification is typically post hoc, if it exists at all.

ThreePod, called Morphix in the user interface, asks a different question: can the generation process itself be made auditable? The system treats each model output as a proposal. A deterministic layer checks the proposal, commits only what passes, and records the evidence. The text route combines atomic action planning, guarded Three.js code generation, syntax checking, headless-browser inspection, and live geometric validation. The image route adds silhouette-based

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

alignment, observed-versus-inferred surface accounting, and Khronos GLB validation. The print route converts an approved result into a simulated, layer-locked path with one authoritative toolhead, material and time budgets, and replayable checkpoints.

The paper makes four contributions:

- a proposal-and-commit architecture for text-to-scene generation, in which schema, grounding, syntax, rendering, and geometry checks gate every committed block;
- an image-to-3D route that reports visible-side agreement and separates observed from inferred surface area instead of hiding the hidden side;
- a printer-independent, layer-locked print protocol that verifies toolhead continuity, layer order, bounds, budgets, and termination;
- a local, inspectable implementation and an evidence bundle for every run, with an offline test suite and named benchmark results.

We do not claim photorealism, open-world generalisation, or state-of-the-art shape quality. The system is a working prototype with a small evaluation. Its contribution is the verification and evidence structure, not a new generative model.

II. RELATED WORK

A. Generative 3D Models

DreamFusion and Magic3D demonstrated that a 2D diffusion prior can supervise a 3D representation [1], [2]. Shap-E and Point-E generate implicit shapes or point clouds conditioned on text or images [3], [4]. Zero-1-to-3 and TripoSR address single-image reconstruction [5], [6]. Hunyuan3D-2 and TRELLIS push shape and texture fidelity with large diffusion models and structured latents [7], [8]. These systems produce impressive assets, but their output is typically a mesh or a radiance field rather than a verified, part-labelled, print-aware object.

B. LLM Code Agents and 3D Tools

BlenderGPT uses a large language model to generate Blender Python code [9]. SceneCraft synthesises 3D scenes as Blender code with an LLM agent [10]. These systems make the output editable and inspectable, which is a major advantage over an opaque mesh. Their usual success criterion is that the program runs, not that the resulting geometry satisfies declared properties.

OpenSCAD and CadQuery provide parametric, exact solid modelling [11], [12]. Cura and PrusaSlicer convert a mesh into a machine-specific toolpath [13], [14]. These tools are strong where ThreePod is weak: exact geometry and manufacturing execution. ThreePod sits upstream of them and does not replace them.

C. Verification, Exchange, and Provenance

The glTF 2.0 specification and the Khronos validator provide a concrete exchange check for GLB assets [15], [16]. They do not, by themselves, verify printability: wall thickness, overhang, support, and machine parameters cannot be inferred from a mesh alone. A general-purpose exchange check is necessary but not sufficient.

Observed-versus-inferred provenance is also uncommon. Single-image systems usually report a visual result and leave the hidden side implicit. ThreePod reports the share of surface area supported by the photograph and the share produced by a prior. This is a modest but useful form of honesty.

The positioning is deliberately narrow and deep. Hosted generators and frontier research models are stronger on photorealism and open-world generalisation. ThreePod is stronger on per-step verification, editable output, observed/inferred provenance, local operation, and a print-aware path. It is weak where the task requires broad world knowledge or photorealistic synthesis, and it does not replace a CAD kernel or a slicer.

III. SYSTEM DESIGN

A. Overview

ThreePod has three workflows that share one evidence model: text-to-scene, image-to-3D, and print-path generation. Fig. 1 shows the end-to-end structure. A user request creates a run. A planner or reconstruction route proposes content. A deterministic layer checks the proposal. Only a passing proposal is committed. The committed result is exported as editable code or a part-labelled object, and an approved result can be converted into a print path. Every step writes to the run folder.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

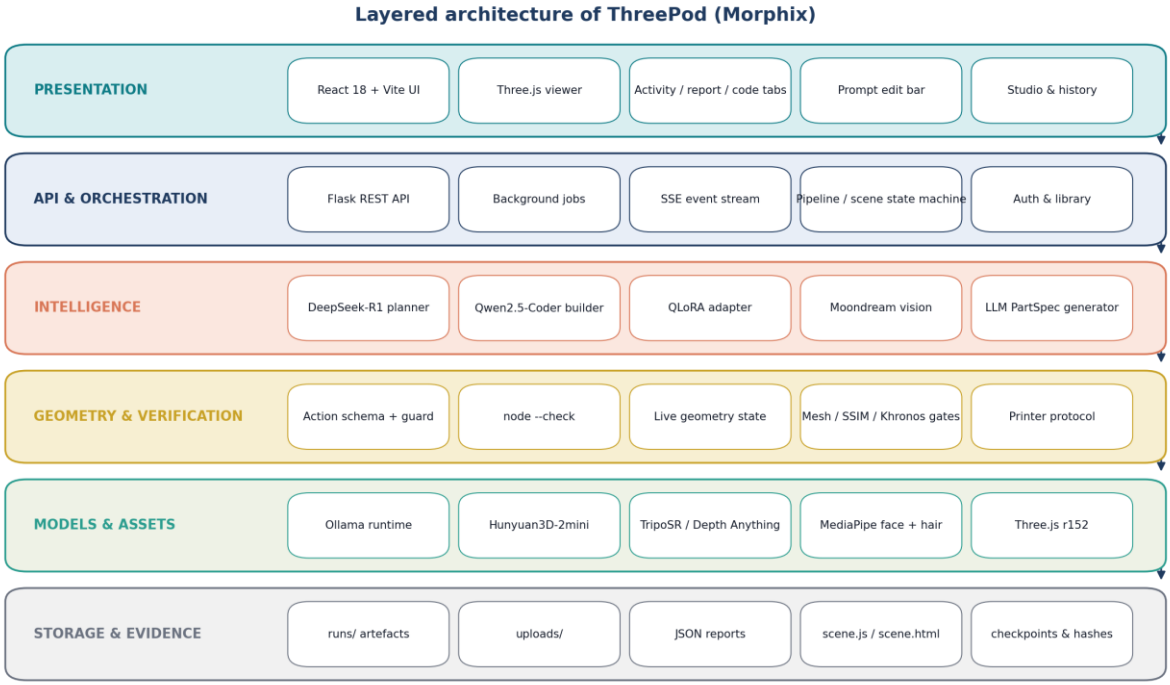


Fig. 1. ThreePod end-to-end architecture. Generative proposals pass through deterministic checks before commit; the same run folder stores code, reports, meshes, and print paths.

B. Text-to-Scene Route

The planner converts a prompt into a sequence of atomic actions. Each action has a type, required parameters, and a target object identifier. The action is checked against a JSON schema before any code is generated. The builder then translates one action into one JavaScript block for a Three.js scene. The block must reference the action's parameters; this grounding check rejects code that ignores the requested dimensions, colours, or positions.

The guard checks for forbidden geometry and degenerate markers before the block is compiled. It rejects empty objects, non-finite dimensions, invalid primitive parameters, and code that bypasses the scene API. The block is then checked with node --check. A syntax error is a blocking failure, not a warning.

A headless Chromium browser loads the candidate scene through Playwright [21]. The validator inspects the live scene graph and checks presence, finiteness, solidity, world bounds, declared contacts and supports, and suspicious floating. A block is committed only when the scene reaches the completion state: every required element is present and rendered, the geometry is valid, there is no fatal runtime error, and no critical defect is unresolved. The state machine moves through planned, building, rendering, validating, and complete or failed states. Failed blocks are retried within a bound; if the model still fails, a deterministic fallback can render the requested primitive. The fallback is validated like any other block and is recorded separately from model success.

C. Image-to-3D Route

The image route begins with subject segmentation and cropping. A bounded vision description extracts only the attributes needed for routing and colour, rather than asking an open-ended model to invent a 3D structure. The system selects a portrait, object, or approximation route.

For object reconstruction, ThreePod uses a learned shape model, Hunyuan3D-2mini, or a recipe fit when the learned route is unavailable [7]. The mesh is repaired by retaining the main connected component and removing fragments. Alignment searches 24 yaw angles, 4 pitch angles, and 5 fields of view, for 480 candidate views; the best and median visible-side silhouette IoU are both recorded. Colour is projected from the visible side and propagated to the hidden side. The object is split into named parts and exported as an editable object.json plus a GLB.

Every GLB is checked with the Khronos validator [16]. The report records the IoU, the status, and the observed-versus-inferred surface split. An IoU below 0.80 is not presented as a full reconstruction; it is labelled an approximation and shown with a warning. Severe failures are labelled failed. This is a deliberate choice: a low-fit result is still useful evidence, but it is not promoted to a success.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

D. Print-Path Route

The print protocol has six operations: move_to, deposit_segment, deposit_point, complete_path, complete_layer, and complete_object. Every deposit must start at the current toolhead position. Every deposit must lie on the active layer. Material, segment, and time budgets must remain non-negative. The path must end with complete_object. A checkpoint records the toolhead, layer, material usage, and segment count.

The validator checks layer order, toolhead continuity, bounds, budget compliance, and termination. The protocol is printer-independent and simplified: it is not a replacement for a slicer. Its value is that it is verifiable and replayable at the level of the generative pipeline. A slicer can produce a better machine path; ThreePod can prove that the path it produced did not teleport, skip a layer, or exceed its budget.

E. Evidence Model

A run is a self-describing folder containing the prompt or image, the actions, the generated code or mesh, the validation report, the trace, the print path, and checkpoints. The database stores accounts and ownership, but the run folder remains inspectable if the database is lost. This is the practical form of reproducibility: every run can be audited after the fact, even if it cannot be regenerated bit-for-bit.

IV. IMPLEMENTATION

The backend is a Flask API with SQLite-backed accounts and ownership. The client is a React and Three.js application [20]. Local models are served through Ollama [22]. The system runs without a paid generation API, and user photographs remain on the local filesystem. Docker Compose is used for reproducible deployment.

The text builder uses Qwen2.5-Coder models [17]. A QLoRA adapter [18], [19] was trained for the 7B model with rank 16, alpha 32, dropout 0.05, 125 steps, one epoch, and 2,000 examples. The final training loss was approximately 0.10, and the adapter is 161.5 MB. The 3B model is used when latency and memory matter; the 7B model is used when target accuracy matters.

The repository contains 200 files and 66,312 lines of code, including 67 test files. The test suite covers unit, integration, visual-regression, and live-model layers; the offline layer is deterministic and does not call a model. Playwright and the Khronos validator are external oracles.

V. EVALUATION

A. Protocol

We report five pieces of evidence. The offline suite measures deterministic behaviour. The scene benchmark measures completion and declared geometry on ten prompts with seed 11. The image study measures visible-side silhouette agreement and GLB validity on four research assets: a portrait, a building, an airplane, and a statue. The production route measures seven named inputs through the learned image route. The model comparison measures 3B versus 7B code models on 30 held-out patch prompts, on CPU only.

The benchmark is descriptive, not a controlled comparison against every system discussed above. It is designed to expose failure modes and to make the evidence traceable. Table I summarises the measured evidence; the following subsections give the protocol and the caveats. All numbers below come from named runs; they are not estimates.

B. Offline Test Suite

The suite collects 766 tests. It passes 723, skips 2, deselects 41 live or visual tests, and fails 0 in 265.07 seconds. The 41 deselected tests are not counted as passes. The result supports the deterministic behaviour of the pipeline, not the quality of the generative model.

C. Scene Benchmark

All ten prompts complete. The validator runs 401 geometry checks and records zero failures. The mean wall-clock time is 39.29 seconds, the median is 23.89 seconds, the minimum is 10.78 seconds, and the maximum is 186.31 seconds. The distribution is heavy-tailed: a warm run is fast, while a cold or retried run can take minutes.

Fig. 2 shows the contact sheet. Completion is not quality. Some scenes are recognisable and some are sparse. The checks prove that required objects are present, finite, solid, in bounds, and consistent with declared relations; they do not prove that the scene is beautiful or that it matches every human interpretation of the prompt.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

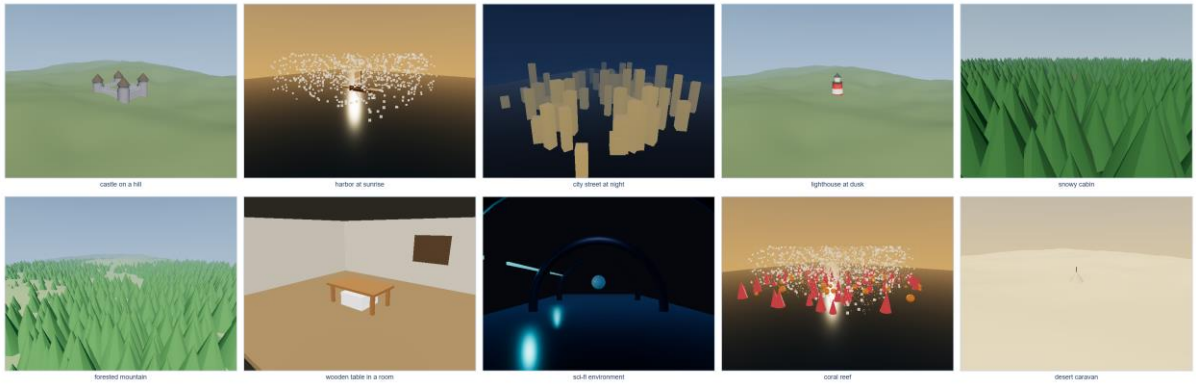


Fig. 2. Contact sheet of the ten scene-benchmark outputs. Completion and geometry checks do not guarantee visual richness.

D. Image-to-3D Results

All four research assets export a GLB with zero Khronos errors and zero warnings. The visible-side IoU is 0.9565 for the portrait, 0.9640 for the building, 0.6016 for the airplane, and 0.9086 for the statue at a 30-degree yaw. The observed surface area ranges from 32.09% to 43.43%; the remainder is inferred. The airplane is a reported failure case: it is thin, elongated, and pose-ambiguous from a single view. Its result is shown with a warning rather than hidden.

The production route covers seven inputs and reaches IoU values from 0.781 to 0.953. A representative car run reconstructs 82,270 vertices and 165,500 triangles in 149.1 seconds with IoU 0.8938, quality accepted, and a seen-vertex fraction of 0.118. Its parts are colour clusters (“Blonde part”, “Black part”, “Beige part”), not semantic labels. This is a known limitation.

Fig. 3 visualises the IoU and the observed/inferred split. The split is the more important number. Even the best asset, the building, has only 39.07% of its surface area observed in the photo; the other 60.93% is inferred. A single-image system that reports only IoU hides this fact.

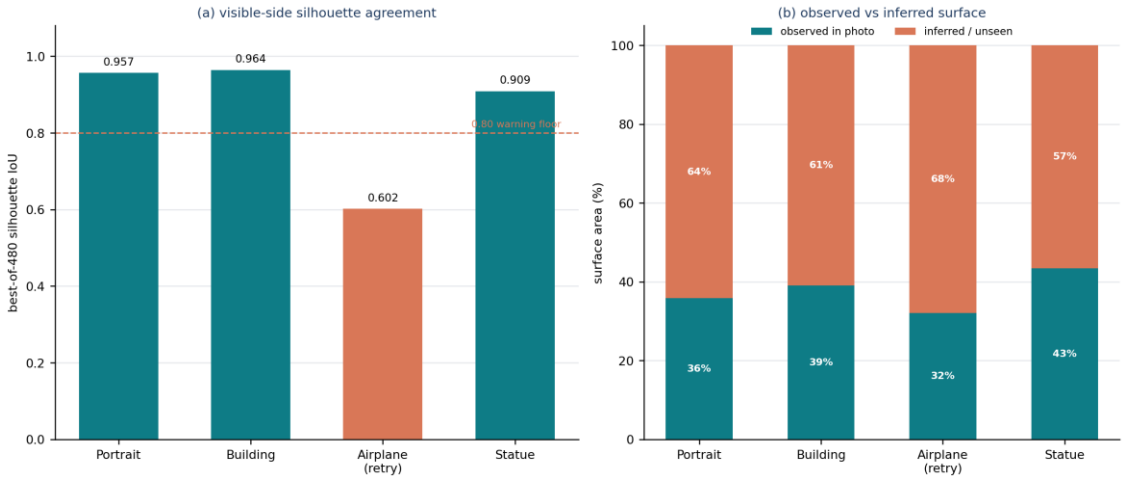


Fig. 3. Visible-side silhouette IoU and observed-versus-inferred surface area for the four research assets. The airplane is a reported failure case (IoU 0.6016).

E. Model Comparison

On 30 held-out patch prompts, the 3B model has a warm median latency of 3.4932 seconds and a peak memory of 2.09 GB. It produces strict JSON on 30/30 prompts, a valid patch schema on 24/30, and the correct target on 23/30. The 7B model has a warm median of 6.3897 seconds and a peak memory of 4.95 GB. It produces strict JSON on 28/30, a valid patch schema on 27/30, and the correct target on 27/30. Cold load is 87.74 seconds for 3B and 334.71 seconds for 7B.

The 7B model is safer on target values and patch structure, but it is roughly 1.8 times slower when warm, uses 2.4 times the memory, and pays a much larger cold-start cost. The 3B model is preferable for interactive use; the 7B model is preferable when the builder must hit a specific parameter. This is a deployment trade-off, not a claim that one model is universally better.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510143

TABLE I
MEASURED EVIDENCE. THE NUMBERS ARE DESCRIPTIVE CASE STUDIES, NOT POPULATION ESTIMATES.

Experiment	Result	Caveat
Offline suite	766 collected; 723 passed, 2 skipped, 41 deselected, 0 failed; 265 s	No live model
Scene benchmark	10/10 complete; 401 checks, 0 failures; 39.29 s mean, 23.89 s median	10 prompts; quality not scored
Research images	4/4 GLB pass; IoU 0.9565, 0.9640, 0.6016, 0.9086	Visible side only
Production route	IoU 0.781–0.953 on 7 inputs	Case study
Qwen 3B vs 7B	3.49 vs 6.39 s warm; 23/30 vs 27/30 target; 2.09 vs 4.95 GB	CPU only; 30 prompts

VI. DISCUSSION AND LIMITATIONS

The results support a narrow claim: a local generative pipeline can be made inspectable and can refuse to commit output that fails declared checks. The scene benchmark shows that the checks scale to ten prompts and 401 assertions without a failure. The image study shows that GLB exchange validity and visible-side agreement can be measured for every asset, including a failure. The model comparison shows that local code models are usable on CPU for small patch tasks, with a clear latency–quality trade-off.

The airplane is the most important negative result. Its best-of-480 IoU is 0.6016, below the 0.80 reconstruction floor. Thin, elongated objects with an ambiguous single-view pose are difficult for silhouette alignment. Reporting the failure is more useful than excluding it.

The fallback is a reliability feature, not a model score. When the builder fails, the guard can render a correct primitive for the requested block. The fallback is recorded separately, so a reader can distinguish model success from pipeline success. This distinction matters: a system can be reliable because it falls back well, while the underlying model remains weak.

The evaluation has clear limits. Ten scene prompts and four research assets are case studies, not a representative benchmark. There are no confidence intervals, no repeated independent samples, and no user study. The print path is simulated; no physical printer was used. The image parts are colour clusters, not semantic segmentation. Three.js is not a CAD kernel, so exact tolerances and manufacturing constraints are outside scope. Security is basic: sessions, hashed passwords, and SQLite, but no multi-tenant hardening, rate limiting, or audit log.

Future work has five concrete directions: a printer adapter for one real machine, a semantic part splitter, a larger independent prompt set, multi-view reconstruction, and a CAD export path. A user study and a comparison against a hosted generator on the same prompts would turn the current case studies into a stronger evaluation.

VII. CONCLUSION

ThreePod is a local-first, verification-first pipeline for prompt- and image-driven 3D modelling and printing. It uses generative models to propose content, but it commits only what passes schema, grounding, syntax, live-browser, geometry, image-quality, and GLB checks. It reports observed versus inferred surface, exports editable parts and code, and converts approved results into a layer-locked print path. The implementation passes 723 of 766 collected tests, completes 10/10 scene prompts with 401 checks and zero failures, and exports four research assets with valid GLBs and IoU from 0.6016 to 0.9640. It is weaker than commercial and frontier research systems on photorealism and open-world generalisation, and it has not been tested on physical hardware. Its contribution is a pipeline that makes generation inspectable, measurable, and print-aware.

REFERENCES

[1]. B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, “DreamFusion: Text-to-3D using 2D diffusion,” in Proc. ICLR, 2023.

[2]. C.-H. Lin et al., “Magic3D: High-resolution text-to-3D content creation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2023, pp. 300–309.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510143

[3]. H. Jun and A. Nichol, "Shap-E: Generating conditional 3D implicit functions," arXiv:2305.02463, 2023.
[4]. A. Nichol et al., "Point-E: A system for generating 3D point clouds from complex prompts," arXiv:2212.08751, 2022.
[5]. R. Liu et al., "Zero-1-to-3: Zero-shot one image to 3D object," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2023, pp. 9298–9309.
[6]. D. Tochilkin et al., "TripoSR: Fast 3D object reconstruction from a single image," arXiv:2403.02151, 2024.
[7]. Tencent Hunyuan3D Team, "Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3D assets generation," arXiv:2501.12202, 2025.
[8]. J. Xiang et al., "Structured 3D latents for scalable and versatile 3D generation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2025.
[9]. BlenderGPT, "Using GPT-4 to generate Blender Python code," GitHub, 2023. [Online]. Available: https://github.com/gd3kr/BlenderGPT
[10]. Z. Yang et al., "SceneCraft: An LLM agent for synthesizing 3D scenes as Blender code," in Proc. Int. Conf. Mach. Learn. (ICML), 2024.
[11]. OpenSCAD, "OpenSCAD: The programmer's solid 3D CAD modeler," 2024. [Online]. Available: https://openscad.org
[12]. CadQuery, "CadQuery: A Python parametric CAD scripting framework," 2024. [Online]. Available: https://cadquery.readthedocs.io
[13]. UltiMaker, "UltiMaker Cura documentation," 2024. [Online]. Available: https://support.ultimaker.com
[14]. Prusa Research, "PrusaSlicer documentation," 2024. [Online]. Available: https://help.prusa3d.com
[15]. Khronos Group, "glTF 2.0 specification," 2024. [Online]. Available: https://www.khronos.org/gltf/
[16]. Khronos Group, "glTF-Validator," 2024. [Online]. Available: https://github.com/KhronosGroup/glTF-Validator
[17]. Qwen Team, "Qwen2.5-Coder technical report," arXiv:2409.12186, 2024.
[18]. E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in Proc. ICLR, 2022.
[19]. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2023.
[20]. Three.js, "Three.js: JavaScript 3D library," 2024. [Online]. Available: https://threejs.org
[21]. Microsoft, "Playwright: Fast and reliable end-to-end testing for the web," 2024. [Online]. Available: https://playwright.dev
[22]. Ollama, "Ollama: Run large language models locally," 2024. [Online]. Available: https://ollama.com

 This work is licensed under a Creative Commons Attribution 4.0 International License