LMBUILD: EVALUATING LLM AGENTS FOR GENERATING BUILDABLE AND FUNCTIONAL STRUCTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM)-based agents are increasingly capable of generating complex 3D structures, with the potential to reshape how objects are designed and realized in the physical world. Yet, producing elegant geometry is fundamentally different from producing objects that can actually be built and perform their intended functions. Existing evaluations largely focus on geometric or visual quality while overlooking structural soundness, functional affordances, and physical realizability. We introduce **LMBuild**, a benchmark for evaluating LLM agents on generating buildable and functional structures. LMBuild represents generated objects as assembled structures comprising meaningful part decompositions, joints, materials, and assembly sequences. To support reproducible evaluation, we provide a unified framework consisting of: (1) an interactive environment in which agents can use tools to retrieve, create, and place components to construct an object; (2) a curated benchmark that repurposes established CAD datasets and augments them with real-world object knowledge from Wikipedia; and (3) a four-level evaluation framework covering structural soundness, functional affordance, design quality, and physical realization. Evaluations across 30 domain-specific models, open-source LLMs, and frontier APIs reveal several intriguing findings: (a) Structural soundness and visual alignment are no longer the primary bottlenecks for frontier closed-source models, while functional affordance and physical operability remain substantially more challenging; (b) stronger models more effectively exploit part creation to construct structure-specific components, whereas weaker models tend to rely on retrieval and can struggle with the additional flexibility introduced by creation; and (c) providing explicit functional specifications substantially improves functional-part completeness, kinematic correctness, and overall physical operability. These results show that generating real-world structures requires capabilities beyond geometric synthesis, including deeper reasoning about functional affordances, mechanics, and physical constraints, as well as the ability to design and create novel components when existing ones are insufficient. We expect LMBuild to provide a foundation for measuring progress and incentivizing research toward agents that generate buildable and functional structures, while helping narrow the gap between open-source communities and frontier labs.

1 INTRODUCTION

The recent release of GPT-6 Astra (OpenAI, 2026b) has sparked an explosion of discussion around how much LLM-based agents could transform 3D structure generation. Beyond their established capabilities in language and code, LLM-based agents are increasingly able to directly manipulate complex 3D assets and scenes (Hu et al., 2024; Sun et al., 2025; Tang et al., 2025b; Liu et al., 2026) and translate language-based specifications into parametric or programmatic 3D designs (Li et al., 2025; Wang et al., 2025b; Alrashedy et al., 2025). These developments lead us to explore a more ambitious possibility: whether LLM agents could reshape not only how structures are digitally designed, but also how they are constructed and realized in the physical world (Kyaw et al., 2025).

However, producing visually plausible geometry is fundamentally different from producing an object that can actually be built and function in the real world. A structure may appear convincing

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.

while remaining invalid, unusable, or impossible to construct. Existing 3D-generation evaluations largely emphasize geometric quality (Poole et al., 2023; Lin et al., 2023), visual consistency (Wang et al., 2023; Maiti et al., 2025; He et al., 2023; Duggal et al., 2025), and semantic alignment (Wu et al., 2024; Zhang et al., 2025), while overlooking critical aspects of physical realization. Yet, we still lack a systematic way to evaluate whether LLM-based agents can leverage their extensive prior knowledge and reasoning capabilities to generate structures that are not only digitally plausible, but also buildable, functional, and aligned with real-world physical constraints.

To this end, we propose **LMBuild**, a first step toward systematically evaluating LLM agents on the generation of buildable and functional structures, while also accommodating domain-specific structure-generation baselines. First, LMBuild provides a unified framework (Figure 1) in which LLM agents are equipped with a standardized agent harness and can interact seamlessly with a 3D environment through a set of construction tools, including retrieving components from a sub-part pool, creating new parts with arbitrary geometry, placing components at desired poses, and refining structures through inspection and revision. Second, as summarized in Table 1, we select 50 high-quality reference targets by repurposing existing 3D datasets containing over 367K data, filtering objects by category and model quality, and augmenting them with real-world knowledge extracted from Wikipedia. This knowledge includes core functional parts, object attributes, joints, and kinematic relationships, which serve as textual anchors for evaluating physically grounded properties. Finally, as shown in Figure 1, we introduce a bottom-up evaluation hierarchy covering four dimensions: (1) structural soundness, assessing whether components form valid and stable 3D assemblies; (2) functional affordance, evaluating whether functional parts, joints, and kinematics support the intended object functions; (3) design quality, assessing decomposition, aesthetics, and overall structural design; and (4) physical realization, evaluating whether the structure can be fabricated with appropriate materials and assembly sequences and ultimately operated or used in the physical world.

We evaluate 30 systems on LMBuild, including 6 closed-source APIs, 13 open-source LLMs, and 11 domain-specific systems, under diverse evaluation settings. Our results show that:

- Structural soundness and visual alignment are no longer primary bottlenecks for frontier closed-source models. Most achieve near-perfect soundness and strong design quality, although open-source models still show room for improvement in aesthetics, and fine-grained visual alignment.
- Functional affordance and physical operability remain the largest capability gaps. All model families struggle with generating a complete list of functional parts, correct joints with kinematics, and artifacts that remain operable under simulation-based evaluation.
- While frontier models can effectively create new components, weaker models rely heavily on retrieval and struggle with the additional reasoning burden introduced by part creation.
- Enriching prompts with desired attributes and functional requirements can force the model to reason, thus improving functional-part completeness, kinematic correctness, and physical operability.

We further show that LMBuild can diagnose failure patterns for open-source LLM agents during structure construction, revealing their gap in part-creation capability as well as persistent challenges in scene-state tracking. Overall, we expect LMBuild to provide a foundation for measuring progress and incentivizing research toward agents capable of generating buildable and functional structures.

2 LMBUILD

In this section, we describe the three core components of **LMBuild**: (1) how we construct an interactive environment for evaluating LLM agents and other systems on generating buildable and functional structures (§ 2.1); (2) how we repurpose existing datasets to support large-scale LMBuild evaluation and illustrate the interactive generation process with a concrete running example (§ 2.2); and (3) how we establish a hierarchical evaluation framework for the generated structures (§ 2.3).

2.1 INTERACTIVE GENERATION OF BUILDABLE AND FUNCTIONAL STRUCTURES

As shown in Figure 1, LMBuild constructs an interactive environment in which agents are equipped with a manually designed harness (detailed in AppendixA.1) and instructed to follow user-provided construction specifications. The agents interact with a set of tools (detailed in AppendixA.2) to generate decomposed and buildable structures, determine appropriate joints and materials for individual components, and specify a feasible assembly sequence for real-world construction.

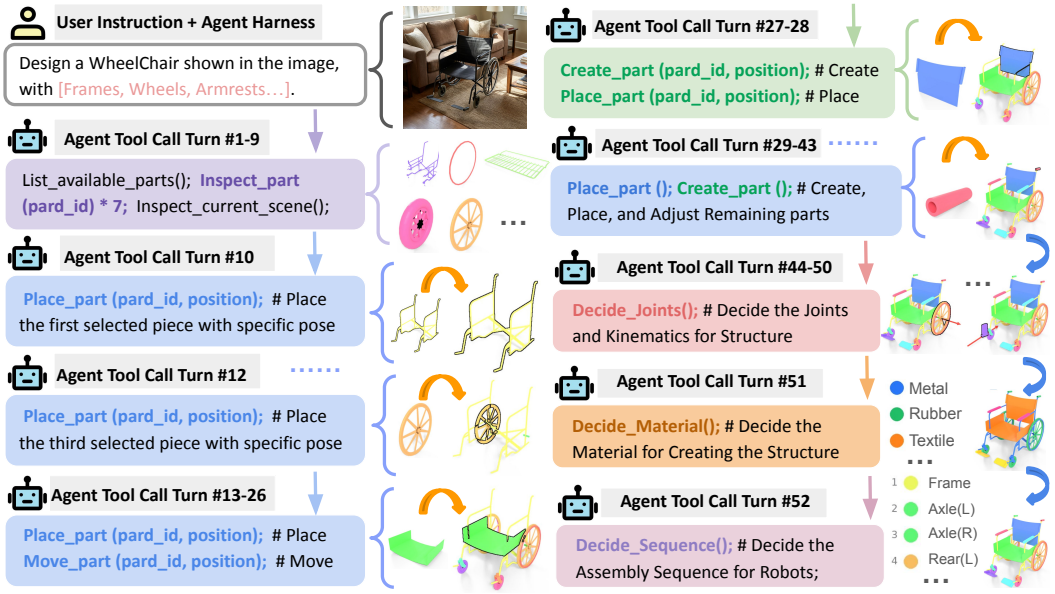


Figure 1: **LMBuild Framework** (Using real example produced by GPT-6 Astra). Given a text prompt and conditional image, the builder LLM agent uses tools to inspect and select existing parts, create new parts, and place and adjust components to construct the wheelchair. It then specifies joints, materials, and an assembly sequence to support kinematics and physical-world realization.

Baseline Interaction Protocol. LMBuild simulates a real-world setting in which users can "speak structures into existence" Kyaw et al. (2025). The user provides only a target structure label and a reference image, without geometric specifications, part-level instructions, or intermediate feedback. The agent then autonomously constructs the object in a single interaction round through tool calls (Figure 1), either retrieving components from a candidate pool containing at least three times the required number of parts or creating new parts when needed. It selects or creates components, places and adjusts them, and specifies the final pose. For domain-specific systems, we preserve their native input and output protocols rather than introducing unsupported capabilities. When a system produces geometry without articulation information, we apply Particulate Li et al. (2026) to infer part-level articulation, joints, and kinematic constraints for evaluation under LMBuild metrics.

Part Retrieval and Creation. The baseline allows agents to **either** retrieve existing components or create new ones when needed. To isolate the contribution of each mechanism, we additionally evaluate two constrained settings: **retrieval-only**, where agents must construct the target structure entirely from the provided component pool, and **creation-only**, where retrieval is disabled and every required component must be generated by the agent. Comparing these settings with the baseline reveals how much agents depend on existing parts versus their ability to design novel components.

Multi-Round Iterative Revision. LMBuild also evaluates whether agents can iteratively improve their constructions using feedback. After each round, the environment returns a six-view rendering and a rule-based diagnostic report covering collisions, disconnected components, stability, joint issues, and missing functional roles, materials, etc. The agent receives this feedback together with its previous interaction trajectory to revises the current structure. We evaluate budgets of $K \in 1, 2, 3$ rounds to measure how effectively agents use this feedback for iterative refinement.

Prompting with Fine-Grained Descriptions. Beyond the baseline category label and reference image, LMBuild evaluates whether richer task specifications improve structure generation. We augment the prompt with fine-grained descriptions of salient attributes and functional components, providing explicit guidance about what parts should be present and how they should behave or be operated. Comparison with the baseline reveals whether failures stem from insufficient specification of user intent or from limitations in translating detailed requirements into buildable structures.

Table 1: Statistics of the source datasets repurposed for **LMBuild**, including original dataset size, objects retained in LMBuild-Core and LMBuild-Full, and average number of Wikipedia-augmented part and affordance annotations obtained. Affordances include functional attributes and kinematic information. We evaluate on the LMBuild-Core split in this work.

Geometry Type	Data Source	Original Size	LMBuild-Core		LMBuild-Full	
			Size	Avg. Parts / Aff.	Size	Avg. Parts / Aff.
LEGO Toy Data	BrickNet (Kulits & Schmid, 2026)	320,808	5	13.80 / 14.60	888	8.79 / 10.82
	BrickComposer (Liu et al., 2026)	9,977	5	10.60 / 11.80	5	10.60 / 11.80
Static Structures	PartNeXt (Wang et al., 2025a)	23,519	6	4.33 / 9.50	581	6.01 / 7.94
Mechanical Structures	Fusion 360 Gallery (Joinable) (Willis et al., 2022)	8,251	3	7.67 / 9.67	28	5.07 / 7.57
Daily Objects / Furniture	Artiverse (Iliash et al., 2026)	5,402	26	6.88 / 9.27	892	5.90 / 7.43
Product-level CAD	ASTRA Quadruped (Cominelli, 2025)	1	1	8.00 / 9.00	1	8.00 / 9.00
	Hubert Robotic Arm (Gildea)	1	1	7.00 / 9.00	1	7.00 / 9.00
	Ploopy Electronics (Ploopy Corporation)	3	3	6.00 / 7.00	3	6.00 / 7.00
Total in LMBuild		367,962	50	7.66 / 9.96	2,399	6.99 / 8.82

2.2 Repurposing Existing Datasets for LMBuild

Evaluating LLM agents for structure generation is substantially more expensive than standard conversational evaluation. Even a simple object with fewer than 20 parts can require over 50 interactions in one construction round, with costs increasing further for complex structures and multi-round revision. Exhaustive evaluation over large 3D datasets is therefore impractical, potentially requiring millions turns of LLM generations. From more than 367K candidate 3D objects, we therefore construct a full benchmark of 2,399 objects and a core set of 50 objects through stratified selection from complementary sources, as summarized in Table 1. Rather than matching the frequency distribution of any individual source dataset, our selection is **coverage-oriented**: we prioritize objects that span diverse construction regimes and capabilities, have verifiable real-world references, and remain tractable for comprehensive evaluation. Due to budget constraints, all experiments in this work are conducted on the core set, while the full benchmark will be made publicly available.

Complementary Geometry Sources. No single existing dataset covers the full range of structures and construction regimes targeted by LMBuild. As summarized in Table 1, we therefore combine several complementary sources. **BrickNet** (Kulits & Schmid, 2026) and **BrickComposer** (Liu et al., 2026) provide LEGO-style structures assembled from reusable part inventories and with unique joints. **PartNeXt** (Wang et al., 2025a) contributes fine-grained part-based objects and static structures, while the **Joinable** (Willis et al., 2022) provides engineering CAD assemblies with explicit part and joint structure. **Artiverse** (Iliash et al., 2026) contributes physically grounded articulated everyday objects with functional parts and kinematic annotations. Finally, open-source product-level CAD from **ASTRA** (Cominelli, 2025), **Hubert** (Gildea), and **Ploopy** (Ploopy Corporation) introduces real engineered systems with documented components and assembly information. Together, these sources span both assembly from reusable part inventories and settings requiring more structure-specific part creation, and cover a broad range of real-world structures.

Data Selection Strategy. Within this combined source pool, we stratify the benchmark object selection along three dimensions. First, we ensure structural and kinematic diversity across six broad families, spanning static load-bearing, hinged mechanisms, sliding mechanisms, rotating systems, serial kinematic chains, and multi-joint articulated mechanisms. Second, we prioritize grounded and recognizable objects whose expected components, physical properties, and functional behaviors can be verified from external Wikipedia data rather than manually invented specifications. Third, we maintain difficulty coverage from simple assemblies to complex articulated systems by diversifying number of parts. The concrete data selection process and example data is released in Appendix A.3.

2.3 Hierarchical Evaluation Framework for LMBuild

Traditional evaluations of generated structures primarily emphasize geometric quality or semantic alignment. In contrast, LMBuild evaluates whether a generated structure is buildable and functional through a hierarchical framework with four complementary dimensions. **Structural Soundness (S)** assesses whether the generated geometry is physically plausible and structurally stable, avoiding



Figure 2: **Left:** Generation examples produced by the LMBuild framework. Colors are used only to visually distinguish decomposed components. **Right:** The generated structures are evaluated using our comprehensive set of evaluation metrics. We briefly illustrate the reference source and general evaluation procedure for each metric; full evaluation details are provided in Appendix A.4.

impossible configurations or severe geometric inconsistencies. **Functional Affordance (A)** evaluates whether the object exhibits the appropriate overall shape, contains the functional parts required by the target category, and specifies joints and kinematics consistent with the intended motion and functionality. **Design Quality (D)** measures whether the generated structure is appropriately decomposed into meaningful components and whether its overall form and appearance align with the reference structure and human design preferences. **Physical Realization (R)** evaluates whether the structure can be instantiated in the physical world, including whether it admits a feasible robotic assembly sequence, provides sufficient material information for fabrication, and remains operable with respect to its intended functionality. As shown in Figure 2 (Right), these four dimensions are further decomposed into twelve concrete sub-metrics, together with their general corresponding evaluation strategies. To aid presentation, all scores are normalized to a common 0–100 scale. To avoid overloading the main text with algorithmic details, we provide the complete evaluation criteria and implementation protocols in Appendix A.4, with illustrative examples in Table 4.

3 Experiments

3.1 Setup

Evaluated Systems. We evaluate 30 systems spanning proprietary APIs, open-weight foundation models, and domain-specific generators, covering diverse model families, generations, scales, and architectures. Proprietary systems include GPT-6 Astra OpenAI (2026b), GPT-5.6 Sol OpenAI (2026a), Claude Opus 5 Anthropic (2026b), Claude Fable 5.1 Anthropic (2026a), Claude Sonnet 5 Anthropic (2026c), and Claude Haiku 4.5 Anthropic (2025). Open-weight models include six Qwen variants across Qwen3, Qwen3-VL, and Qwen3.5 Yang et al. (2025); Bai et al. (2025); Qwen Team (2026), spanning 4B–35B parameters and both dense and MoE architectures (e.g., 30B-A3B and 35B-A3B), together with GPT-OSS-120B OpenAI (2025), Gemma-4-31B-IT Gemma Team et al. (2026), Ministral-3-8B Mistral AI (2025), InternVL3.5-8B/38B Wang et al. (2025c), MiniCPM-V-4.5 Yu et al. (2025), and ERNIE-4.5-VL-28B-A3B ERNIE Team, Baidu (2025). We further compare with domain-specific systems for brick assembly, part-aware generation, and simulation-ready assets, including BrickGPT Pun et al. (2025a), LegoACE Xu et al. (2025), PartCrafter Lin et al. (2025), PartPacker Tang et al. (2025a), Cube3D and CubePart Foundation AI

Table 2: Evaluation results across 30 systems under Baseline Interaction Protocol (§ 2.1), assessed using our hierarchical evaluation metrics (§ 2.3). We use † (P) to denote systems integrated with Particulate to obtain joint information (Appendix A.2). Background colors encode performance on a common 0–100 scale, transitioning from blue (lower performance) through gray to orange (higher performance). Systems unable to produce the required output label are assigned as N/A.

Models / Metrics	Soundness			Affordance			Design			Realization		
	S.1	S.2	S.3	A.1	A.2	A.3	D.1	D.2	D.3	R.1	R.2	R.3
State-of-the-art Closed-Source APIs												
GPT-6 Astra	92.6	100.0	92.9	71.8	63.5	53.0	76.2	79.4	82.8	64.0	73.5	34.7
GPT-5.6 Sol	91.2	100.0	90.3	56.4	64.7	50.9	78.5	77.5	74.3	62.5	82.9	33.8
Claude Opus 5	95.2	100.0	91.5	60.7	63.5	45.4	75.7	76.6	77.5	56.3	76.1	36.2
Claude Fable 5.1	90.4	100.0	89.6	61.1	62.6	53.1	74.7	78.4	77.7	67.3	72.7	35.0
Claude Sonnet 5	85.4	100.0	85.3	55.7	61.6	43.6	77.2	69.4	74.0	72.3	80.9	31.3
Claude Haiku 4.5	90.3	98.3	73.3	39.4	58.4	23.3	73.3	58.4	63.4	56.1	81.1	20.5
Open-Source (Multi-modal) Large Language Models												
Qwen3.5-27B	89.4	98.3	81.8	46.3	53.0	20.8	70.8	53.1	64.1	74.1	79.5	22.1
Qwen3.5-35B-A3B	79.3	94.9	77.8	36.9	50.3	17.6	70.7	49.1	60.0	59.9	78.0	23.5
GPT-OSS-120B	81.1	99.7	70.9	31.2	50.2	16.7	66.1	49.7	54.2	58.1	80.1	22.5
Gemma-4-31B-IT	78.0	92.8	75.8	41.1	46.9	17.3	64.8	52.5	61.2	54.5	77.3	19.4
Ministral-3-8B	72.1	92.6	56.7	23.2	46.4	5.5	60.9	39.6	51.5	28.9	78.9	15.1
InternVL3.5-38B	60.4	87.3	59.8	24.1	33.4	9.9	56.9	40.3	48.6	21.3	68.2	13.1
Qwen3-VL-30B-A3B	72.2	70.2	46.1	24.5	41.7	3.6	52.8	37.8	47.0	32.2	69.6	13.3
MiniCPM-V-4.5	56.0	75.4	41.8	20.6	28.0	2.3	46.2	31.0	33.3	10.0	65.3	9.6
Qwen3-4B-Instruct	85.8	59.8	65.9	23.4	38.2	3.9	45.1	23.5	36.1	37.6	74.1	14.3
Qwen3-VL-32B	58.1	67.0	51.1	23.3	36.8	7.4	46.9	33.5	41.5	32.6	62.2	16.2
ERNIE-4.5-VL-28B-A3B	47.5	59.8	34.9	17.4	28.7	4.0	41.2	32.0	37.7	11.0	53.9	12.8
InternVL3.5-8B	48.5	53.3	46.2	15.5	19.4	2.0	34.1	30.7	32.9	2.9	54.2	5.8
Qwen3-VL-8B	63.2	54.7	54.8	16.4	30.8	4.8	33.4	29.1	32.7	38.0	62.6	12.9
State-of-the-art Domain-Specific Models												
BrickGPT	99.1	100.0	83.9	32.3	40.7	N/A	18.0	16.6	33.6	N/A	N/A	N/A
LegoACE	94.0	100.0	74.8	24.9	32.5	N/A	15.3	11.8	34.3	N/A	N/A	N/A
PartCrafter-15	89.0	89.9	25.3	45.8	27.3	N/A	62.4	50.7	64.4	N/A	N/A	N/A
PartCrafter-8	73.6	93.2	29.0	41.3	20.8	N/A	65.0	49.6	63.6	N/A	N/A	N/A
PartPacker	70.3	97.8	65.6	51.1	25.0	N/A	56.7	66.1	72.0	N/A	N/A	N/A
Cube3D+CubePart	88.6	87.1	52.2	29.7	55.7	N/A	57.0	68.6	63.8	N/A	N/A	28.4
PartCrafter15†(P)	82.5	100.0	19.3	45.8	11.0	2.5	30.6	33.8	54.5	N/A	N/A	N/A
PartCrafter8†(P)	79.9	100.0	27.9	42.9	13.4	4.6	39.2	42.7	59.7	N/A	N/A	N/A
PartPacker†(P)	84.4	100.0	65.3	51.2	16.5	2.2	42.2	60.4	66.8	N/A	N/A	N/A
Cube3D†(P)	78.0	100.0	80.4	47.0	16.6	6.1	43.3	85.7	64.8	N/A	N/A	N/A
PhysX-Anything	58.7	97.8	50.9	23.7	22.8	8.3	49.6	51.8	55.9	N/A	69.6	5.8

Team, Roblox (2025); Zhu et al. (2026), and PhysX-Anything Cao et al. (2025b). Selected part-based generators are additionally augmented with Particulate for joint specification (Table 3).

3.2 Measuring Model Capabilities for Functional Structure Generation

Overall Observations. As shown in Table 2, state-of-the-art closed-source LLMs achieve the strongest overall performance across most metrics, although several open-source and domain-specific systems remain competitive and occasionally surpass them on individual geometric (e.g., Cube3D†(P) on **D.2**) or realization metrics (e.g., Qwen3.5 models on **R.1**). Notably, frontier closed-source models, including GPT-6 Astra, GPT-5.6 Sol, Claude Opus 5, and Claude Fable 5.1, exhibit broadly similar performance across many evaluation dimensions, suggesting that leading model families are converging toward a comparable capability level for generating 3D structures. More broadly, overall performance is associated with stronger general reasoning capabilities, while the competitive performance of several domain-specific systems suggests that targeted training and architectural specialization can partially compensate for weaker general reasoning on structure generation.

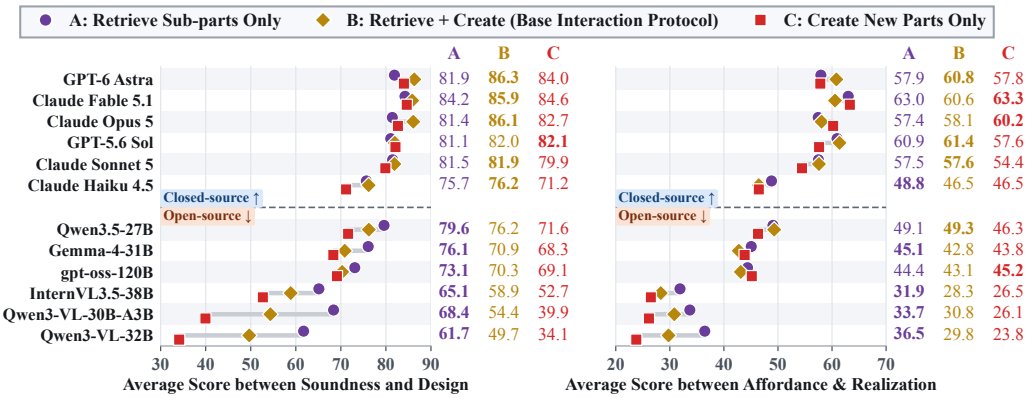


Figure 3: **Model Performance Change** across evaluation metrics under three part-access settings with 25 sampled LMBuild objects. Allowing part creation can hurt weaker models, even when catalog parts remain available, suggesting that the added design freedom increases decision complexity. In contrast, frontier models benefit substantially from creating structure-specific parts, revealing a stronger ability to exploit flexible construction mechanisms rather than relying only on retrieval.

Structural Soundness and Visual Alignment Are No Longer the Primary Bottlenecks Across Structural Soundness, most systems perform reliably in generating interconnected (**S.1**) and collision-free (**S.2**) structures, although performance declines somewhat on structural stability (**S.3**), particularly for domain-specific generators. Similarly, frontier closed-source models already perform strongly in producing reasonable decompositions (**D.1**), aesthetically appealing designs (**D.2**), and reference-aligned structures (**D.3**), suggesting that structural plausibility and visual quality are no longer their primary bottlenecks. Several open-source and domain-specific systems are also competitive on individual design metrics, but a noticeable gap remains overall, leaving room for further improvement in generating structures with better aesthetic quality, stronger structural stability, and more fine-grained visual alignment.

Functional Affordance and Physical Operability Remain the Largest Capability Gaps. Even the strongest systems remain imperfect in generating appropriate functional geometry (**A.1**) and complete functional parts (**A.2**), while joints and kinematics (**A.3**) are particularly challenging: most open-source models score below 25, and even the strongest systems only slightly exceed 50. These limitations carry directly into physical realization. Although models are comparatively more capable of specifying plausible assembly sequences (**R.1**) and selecting materials (**R.2**), simulation-based operability (**R.3**) remains low across all systems, with the best scores reaching only the mid-30s. Together, these results suggest that generating a structurally and visually aligned artifact is substantially easier than producing the functional geometry, complete parts, joint configurations, and kinematics required for the artifact to be fabricated and operate correctly in the physical world.

3.3 Strong Models Create, While Weaker Models Retrieve

We further evaluate LLM agents under three interaction protocols: our baseline setting with both retrieval and part creation, a retrieval-only setting, and a creation-only setting. As shown in Figure 3, model behavior differs substantially across capability levels. Less capable models degrade sharply when required to create all parts themselves. For example, Qwen3-VL-32B achieves an average score of 61.7 across Soundness and Design metrics under retrieval, but drops to 34.1 under creation-only. Even simply introducing part creation alongside retrieval can reduce performance for weaker models, suggesting that additional design freedom also introduces a substantially larger reasoning and decision-making burden. In contrast, frontier models exhibit the opposite trend. They more frequently choose to create their own components when both tools are available, and maintain comparable or even stronger performance when retrieval is removed entirely. For these models, retrieval-only can instead become the more restrictive setting, as the available component pool may constrain their ability to construct geometry tailored to the target structure. This reveals an intriguing capability transition: retrieval helps weaker models by reducing the design space, whereas creation

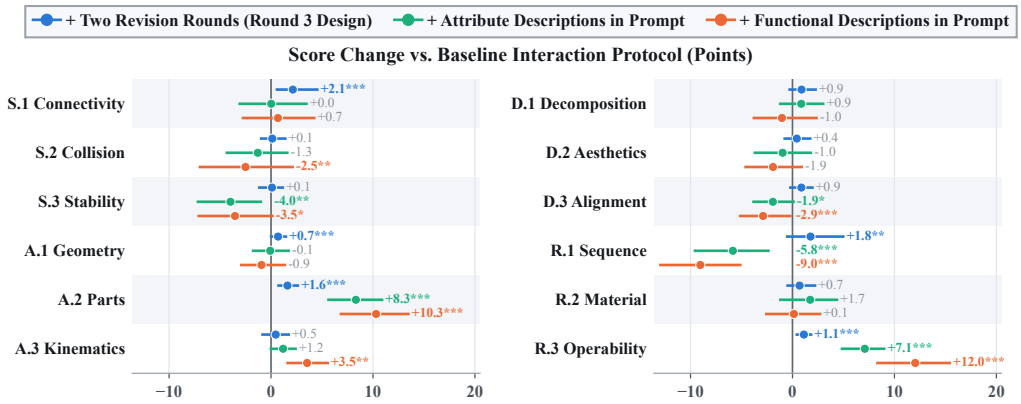


Figure 4: **Average performance change** across models from (1) additional revision rounds, (2) explicit attribute requirements, and (3) explicit functional requirements. Revision yields small but consistent gains, while explicit prompts substantially improve functional-part completeness, kinematics, and especially operability. **Bars:** 95% bootstrap confidence intervals over models; **Stars:** Wilcoxon signed-rank test over episodes (*$p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

allows stronger models to exploit their generative and structural reasoning capabilities. Part creation therefore appears not merely as an additional tool, but as a capability that increasingly becomes beneficial as the underlying model becomes stronger.

3.4 Improving Capabilities with Iterative Revision and Prompting

As shown in Figure 4, we evaluate two common strategies for improving buildable and functional structure generation: (1) allowing models to inspect their outputs, receive error reports, and iteratively revise the structure; and (2) enriching the target description with explicit functional attributes and requirements to encourage deeper reasoning about the underlying structure.

Iterative Revision Provides Only Modest Gains. Allowing models to inspect and revise their structures over three rounds leads to steady but limited improvement, with overall gains of only about 2%. Because agents can already interact with and inspect the environment under the baseline protocol, additional revision rounds mainly provide more opportunities to retry rather than addressing the underlying reasoning limitations required for functional structure generation.

Explicit Functional Specifications Produce Substantial Improvements. In contrast, enriching a minimal prompt with explicit attributes and functional requirements grounded in Wikipedia produces substantially larger gains on the dimensions where current models struggle most. Functional-part completeness improves by 10.3 points and physical operability by 12.0 points ($p < 0.001$), with additional improvements in kinematic correctness. These results indicate that a substantial fraction of baseline failures can be reduced when functional requirements are made explicit. Current agents therefore appear considerably more reliable at translating specified functional constraints into structure than at inferring all such requirements from a minimal object label alone.

Implication. Together, these results suggest that improving task specification can be more effective than increasing the revision rounds. A promising direction is therefore to develop agents that can more reliably infer, retrieve, or elicit the functional requirements of an object before construction, reducing the need for users to manually provide detailed part-level and kinematic specifications.

3.5 LMBuild Identifies Failure Patterns of Evaluated LLMs

Open Models Exhibit a Part-Creation Gap. Across Part Creation calls, closed-source models succeed on 95.7% of attempts, compared with only 51.7% for open-source models, with Qwen3.5-27B (89.4%) as the only close exception. Most failures fall into two recurring categories. First, models violate CAD-kernel conventions, such as listing polygon vertices clockwise. Second, models

generate invalid program structures by nesting nodes instead of referencing named nodes, using numeric node IDs, or referring to undefined nodes. These errors suggest that much of the creation gap stems from weak adherence to tool-specific conventions and program structure, rather than purely from deficiencies in geometric or spatial reasoning.

Open Models Struggle with Scene-State Tracking. Beyond part creation, open-source models frequently lose track of the evolving scene state. They refer to parts that were never placed at a rate of 7.2 errors per 100 calls, compared with 0.3 for closed-source models, they often attempt to attach, join, or assign properties to parts whose creation had already failed. They also redundantly add existing entities, including repeatedly placing parts instead of moving them and re-adding existing joints, at 3.2 errors per 100 calls versus 0.5 for closed-source models. Additional interface errors include invalid material names, malformed arguments, exhausted inspection calls, and even invoking part names as tools. Together, these failures reveal a broader limitation in maintaining and updating an accurate internal representation of the environment throughout long-horizon construction.

4 Related Work

3D Generation with Language Models. Language models increasingly support natural-language 3D creation by generating procedural scripts and scene blueprints (Hu et al., 2024; Sun et al., 2025; Tang et al., 2025b) or parametric CAD operation sequences (Khan et al., 2024; Li et al., 2025). Visual feedback further enables training and iterative refinement of generated geometry (Huang et al., 2024; Wang et al., 2025b; Alrashedy et al., 2025). These methods connect language understanding with programmatic 3D modeling, but generally focus on producing editable or visually correct geometry rather than structures that can be physically assembled and function as intended.

Part-Aware Generation and Physical Assembly. Recent work represents objects as structured collections of parts (Lin et al., 2025; Tang et al., 2025a), predicts joints and articulation (Willis et al., 2022; Li et al., 2026), and incorporates physical attributes such as scale, materials, and motion (Le et al., 2025; Cao et al., 2025a;b). Complementary approaches target stable brick construction, assembly programs, component placement, and robotic execution (Pun et al., 2025b; Kulits & Schmid, 2026; Liu et al., 2026; Goldberg et al., 2025). Together, these works motivate evaluating part structure, physical behavior, and whether the completed object can realize its intended function.

Evaluation of 3D Generation. Existing 3D evaluation primarily measures visual quality and text alignment (He et al., 2023), often extended with multimodal judgments of appearance, surface quality, and human preference (Wu et al., 2024; Maiti et al., 2025), or finer-grained geometric and semantic consistency checks (Duggal et al., 2025). For assembled structures, prior work additionally evaluates stability, component selection, placement, and construction success (Pun et al., 2025b; Liu et al., 2026; Goldberg et al., 2025). LMBuild complements these efforts by jointly evaluating soundness, affordance, design, and physical realization across heterogeneous objects using geometric checks, grounded component assessments, simulation-based motion tests, and VLM judgments.

5 Conclusion

In this work, we introduce LMBuild, a benchmark for evaluating LLM-based agents on the generation of structured 3D assemblies across four dimensions: structural soundness, functional affordance, design quality, and physical realization. On the LMBuild-Core benchmark, the evaluated systems perform substantially better on structural and visual criteria than on functional-part completeness, kinematic correctness, and simulation-based operability. These results highlight persistent challenges in translating object specifications into coherent components, joints, and assembly plans, as well as in creating task-specific parts when suitable components are unavailable. We hope LMBuild provides a reproducible foundation for studying these capabilities and for developing agents that generate increasingly functional and physically plausible structures. Bridging these benchmark results to real-world fabrication and operation will require further validation involving manufacturing constraints, physical assembly, and functional testing.

AI USE STATEMENT

We used large language models (LLMs) as writing assistants to improve the clarity, grammar, and presentation of the manuscript. We also used AI-based coding agents to assist with evaluation-environment construction, code refactoring, debugging, and code cleaning. All research decisions, experimental designs, implementations, analyses, and reported results were reviewed and verified by the authors. AI tools were not used to determine the scientific conclusions of this work.

ETHICS STATEMENT

This work aims to support more transparent, accessible, and reproducible evaluation of LLM agents for real-world structure generation. By developing an open evaluation framework, we hope to lower the barriers to systematically assessing model capabilities beyond purely textual benchmarks. In particular, our benchmark emphasizes whether generated structures are not only plausible in appearance but also buildable and functionally valid under real-world constraints. We believe that open evaluation infrastructure can facilitate more rigorous comparison across models, encourage reproducible research, and contribute to the broader open-source AI ecosystem. At the same time, benchmarks can only capture a subset of the requirements involved in deploying AI-generated designs in real-world settings. Performance on LMBuild should therefore not be interpreted as sufficient evidence that a model is ready for safety-critical engineering, construction, or other physical-world applications without appropriate domain-specific validation and human oversight.

REPRODUCIBILITY STATEMENT

We provide representative data samples and evaluation materials in the supplementary materials to facilitate inspection and reproduction of our methodology. We will open-source the complete codebase, data, evaluation environment, evaluation scripts, model trajectories from the reported experiments, and all the necessary artifacts required to reproduce our results. We aim to make the full evaluation pipeline transparent and executable so that future work can independently verify our findings, evaluate additional models, and extend LMBuild for constructing more powerful agents.

REFERENCES

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating cad code with vision-language models for 3d designs. In International Conference on Learning Representations, volume 2025, pp. 52236–52262, 2025.

Anthropic. Introducing claude haiku 4.5. https://www.anthropic.com/news/claude-haiku-4-5, October 2025. Accessed September 2026.

Anthropic. Claude fable 5.1. https://platform.claude.com/docs/en/models/fable-5-1/overview, 2026a. Accessed September 2026.

Anthropic. Introducing claude opus 5. https://www.anthropic.com/news/claude-opus-5, July 2026b. Accessed September 2026.

Anthropic. Introducing claude sonnet 5. https://www.anthropic.com/news/claude-sonnet-5, June 2026c. Accessed September 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report. arXiv preprint arXiv:2511.21631, 2025.

Ziang Cao, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-3d: Physical-grounded 3d asset generation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), Advances in Neural Information Processing Systems, volume 38, Main Conference, pp. 93771–93784. Curran Associates, Inc., 2025a. doi: 10.52202/085713-3136. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/86beeac1799c083e4597d4aa17385a25-Paper-Conference.pdf.

Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-Anything: Simulation-ready physical 3d assets from single image. arXiv preprint arXiv:2511.13648, 2025b.

Federico Cominelli. ASTRA: Quasi-direct-drive spinal quadruped. Open-source hardware and software repository, 2025. URL https://github.com/fedecomi04/opensource-spinal-quadruped. Accessed: 2026-09-17.

Shivam Duggal, Yushi Hu, Oscar Michel, Aniruddha Kembhavi, William T Freeman, Noah A Smith, Ranjay Krishna, Antonio Torralba, Ali Farhadi, and Wei-Chiu Ma. Eval3d: Interpretable and fine-grained evaluation for 3d generation. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13326–13336. IEEE, 2025.

ERNIE Team, Baidu. ERNIE 4.5 Technical Report. https://ernie.baidu.com/blog/publication/ERNIE_Technical_Report.pdf, 2025.

Foundation AI Team, Roblox. Cube: A roblox view of 3d intelligence. arXiv preprint arXiv:2503.15475, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, et al. Gemma 4 Technical Report. arXiv preprint arXiv:2607.02770, 2026.

Aidan Gildea. Hubert: 3-dof robotic manipulator. Open-source hardware and software repository. URL https://github.com/Aidan-Gildea/robot-arm-3dof. Accessed: 2026-09-17.

Andrew Goldberg, Kavish Kondap, Tianshuang Qiu, Zehan Ma, Letian Fu, Justin Kerr, Huang Huang, Kaiyuan Chen, Kuan Fang, and Ken Goldberg. Blox-net: Generative design-for-robot-assembly using vlm supervision, physics simulation, and a robot with reset. In 2025 IEEE International Conference on Robotics and Automation (ICRA), pp. 15493–15500. IEEE, 2025.

Yuze He, Yushi Bai, Matthieu Lin, Wang Zhao, Yubin Hu, Jenny Sheng, Ran Yi, Juanzi Li, and Yong-Jin Liu. T^3 bench: Benchmarking current progress in text-to-3d generation. arXiv preprint arXiv:2310.02977, 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.

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In European Conference on Computer Vision, pp. 297–314. Springer, 2024.

Denys Iliash, Jiayi Liu, Egor Fokin, Qirui Wu, Ali Mahdavi-Amiri, Manolis Savva, and Angel X. Chang. Artiverse: A diverse and physically grounded dataset for articulated objects. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8932–8942, 2026.

Mohammad S Khan, Sankalp Sinha, Talha U Sheikh, Didier Stricker, Sk A Ali, and Muhammad Z Afzal. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. Advances in Neural Information Processing Systems, 37:7552–7579, 2024.

Peter Kulits and Cordelia Schmid. Bricknet: Graph-backed generative brick assembly. arXiv preprint arXiv:2604.22984, 2026.

Alexander Htet Kyaw, Miana Smith, Se Hwan Jeon, and Neil Gershenfeld. Speech to reality: On-demand production using natural language, 3d generative ai, and discrete robotic assembly. In Proceedings of the ACM Symposium on Computational Fabrication (SCF '25), pp. 16:1–16:12. Association for Computing Machinery, 2025. doi: 10.1145/3745778.3766670.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In International Conference on Learning Representations, volume 2025, pp. 17578–17602, 2025.

Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. CAD-Llama: Leveraging large language models for computer-aided design parametric 3d model generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18563–18573, June 2025.

Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-forward 3d object articulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 27708–27718, 2026.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3d content creation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 300–309, 2023. doi: 10.1109/CVPR52729.2023.00037.

Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Feng Yiqiang, Yadong Mu, and Katerina Fragkiadaki. PartCrafter: Structured 3d mesh generation via compositional latent diffusion transformers. In Advances in Neural Information Processing Systems, volume 38, 2025. doi: 10.52202/085713-1189.

Jiateng Liu, Bingxuan Li, Zhenhailong Wang, Rushi Wang, Kaiwen Hong, Cheng Qian, Jiayu Liu, Denghui Zhang, Katherine Driggs-Campbell, Manling Li, et al. Brick-composer: Using mllms for assembly with diverse bricks. arXiv preprint arXiv:2606.05445, 2026.

Shalini Maiti, Lourdes Agapito, and Filippos Kokkinos. Gen3deval: Using vllms for automatic evaluation of generated 3d objects. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18552–18562. IEEE, 2025.

Mistral AI. Introducing mistral 3. https://mistral.ai/news/mistral-3/, December 2025. Includes Ministral 3 model family.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026a. Accessed September 2026.

OpenAI. GPT-6 Astra: A new generation of intelligence. https://openai.com/index/gpt-6-astra/, 2026b. Accessed September 2026.

Ploopy Corporation. Ploopy open-source electronics. Open-source hardware repositories. URL https://github.com/ploopyco. Includes Ploopy Mouse, Classic 2 Trackball, and Headphones. Accessed: 2026-09-17.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3d using 2d diffusion. In International Conference on Learning Representations (ICLR), 2023.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 14798–14809, October 2025a.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In 2025 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 14798–14809. IEEE, 2025b.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3D-GPT: Procedural 3d modeling with large language models. In 2025 International Conference on 3D Vision (3DV), pp. 1253–1263, March 2025. doi: 10.1109/3DV66043.2025.00119.

Jiaxiang Tang, Ruijie Lu, Zhaoshuo Li, Zekun Hao, Xuan Li, Fangyin Wei, Shuran Song, Gang Zeng, Ming-Yu Liu, and Tsung-Yi Lin. Efficient part-level 3d object generation via dual volume packing. In Advances in Neural Information Processing Systems, volume 38, 2025a. doi: 10.52202/085713-0914.

Song Tang, Kaiyong Zhao, Lei Wang, Yuliang Li, Xuebo Liu, Junyi Zou, Qiang Wang, and Xiaowen Chu. UnrealLLM: Towards highly controllable and interactable 3d scene generation by llm-powered procedural content generation. In Findings of the Association for Computational Linguistics: ACL 2025, pp. 19417–19435. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.findings-acl.994.

Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding. In Advances in Neural Information Processing Systems, 2025a. Datasets and Benchmarks Track.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-cad generation through infusing visual feedback in large language models. arXiv preprint arXiv:2501.19054, 2025b.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265, 2025c.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In Advances in Neural Information Processing Systems, volume 36, 2023.

Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15828–15839. IEEE, 2022.

Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v (ision) is a human-aligned evaluator for text-to-3d generation. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 22227–22238. IEEE, 2024.

Hao Xu, Yuqing Zhang, Yiqian Wu, Xinyang Zheng, Yutao Liu, Xiangjun Tang, Yunhan Yang, Ding Liang, Yingtian Liu, Yuanchen Guo, Yanpei Cao, and Xiaogang Jin. LegoACE: Autoregressive construction engine for expressive lego® assemblies. In Proceedings of the SIGGRAPH Asia 2025 Conference Papers, pp. 40:1–40:11. ACM, 2025. doi: 10.1145/3757377.3763881.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, Bokai Xu, Junbo Cui, Yingjing Xu, Liqing Ruan, Luoyuan Zhang, Hanyu Liu, Jingkun Tang, Hongyuan Liu, Qining Guo, Wenhao Hu, Bingxiang He, Jie Zhou, Jie Cai, Ji Qi, Zonghao Guo, Chi Chen, Guoyang Zeng, Yuxuan Li, Ganqu Cui, Ning Ding, Xu Han, Yuan Yao, Zhiyuan Liu, and Maosong Sun. MiniCPM-V 4.5: Cooking efficient mllms via architecture, data, and training recipe. arXiv preprint arXiv:2509.18154, 2025.

Yujie Zhang, Bingyang Cui, Qi Yang, Zhu Li, and Yiling Xu. Benchmarking and learning multi-dimensional quality evaluator for text-to-3d generation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 18563–18574, October 2025. doi: 10.1109/ICCV51701.2025.01725.

Yiheng Zhu, Kangle Deng, Jean-Philippe Fauconnier, Inaki Navarro, Daiqing Li, Ava Pun, Yinan Zhang, Peiye Zhuang, Xiaoxia Sun, Maneesh Agrawala, Kiran Bhat, and Tinghui Zhou. CubePart: An open-vocabulary part-controllable 3d generator. In ACM SIGGRAPH 2026 Conference Papers, pp. 7:1–7:12, 2026.

A Appendix

A.1 Agent Harness for LMBuild

Overview. LMBuild exposes a common interactive assembly environment to heterogeneous model families through a lightweight model-specific harness. General-purpose LLM (VLM) agents receive the same environment semantics, task specification, tool definitions, action budgets, and multi-round revision protocol. The harness changes only the mechanism by which observations are delivered and actions are returned: closed-source agents interact with the environment through shell commands, whereas locally served open-source models use an OpenAI-compatible chat interface with text-serialized tool calls. Domain-specific structure-generation systems do not support the common tool interface and are therefore evaluated using their native generation pipelines. This separation allows LMBuild to preserve a common task and environment while accommodating substantial differences in model APIs. In particular, the closed- and open-source agent settings expose equivalent task information, although image transport, tool-call serialization, and conversation-state management necessarily differ across interfaces.

Environment tiers. All general-purpose agents are evaluated in one of three environment specifications. We refer to them as Tier A, Tier B, and Tier C for simplicity. **Tier A** permits retrieval and placement of catalog parts only. **Tier B**, our primary baseline interaction protocol, provides the same catalog while additionally allowing the agent to create a new component when no suitable catalog part is available. **Tier C** removes the catalog entirely and requires the agent to create every component before placing it. Except for the corresponding catalog and part-creation instructions, the environment prompt and tool semantics are unchanged across tiers. The environment enforces a maximum of 160 tool calls per episode, including at most 6 calls to `check`, 4 calls to `render`, and 80 placed parts. Part creation is limited to 30 calls in Tier B and 60 calls in Tier C. Both `check` and `render` are explicitly described to the agent as metered operations so that they are used for verification rather than unrestricted search.

Multi-round interaction. An episode may contain multiple revision rounds. Calling `submit` ends the current round. When another revision round remains, the environment returns a structured diagnostic produced by `check` together with condition, isometric, front, and side-view images of the current assembly. The agent is asked to inspect this feedback, revise the same scene, and submit again. A final `submit` returns `episode_done=true`. The assembly state is persistent across rounds; therefore, revisions modify the existing object rather than restarting generation.

A.1.1 Shared LMBuild Agent Prompt

The following system prompt is shared by all general-purpose LLM/VLM agents. The placeholder {`PARTS_SOURCE`} is instantiated according to the tier, after which LMBuild appends the tier-specific part-creation instructions and the generated tool documentation.

Shared environment system prompt

```
You assemble a physical object out of 3D parts by calling tools. The result is judged as a
↪  real, buildable, working product: it must hold together, stand, look like what was
↪  asked for, move where the real object moves, be made of sensible materials, and be
↪  buildable in the order you give.
```

```
WORLD FRAME
- Units are metres. +Z is up. The ground is the plane z = 0. The FRONT of the object faces
↪  -Y (a person sitting on a chair faces -Y). Standing in front of the object (at -Y) and
↪  looking at it, +X is to your right.
- Real-world size matters. Build the object at its real size.

PARTS
{PARTS_SOURCE}
{TOOLS_DOCUMENTATION}
- Every part's local origin is the centre of its bounding box. place_part puts that centre
↪  at `position` and applies `rotation` = [rx, ry, rz] degrees about the world X, then Y,
↪  then Z axes. So an unrotated part placed at z = size_z / 2 rests exactly on the
↪  ground.
- attach moves a part so that one face centre of its bounding box (top, bottom, left,
↪  right, front, back, center) lands on a face centre of another part's box, plus an
↪  optional offset. It is the easiest way to stack parts exactly: attach(child, "bottom",
↪  parent, "top").
- Parts count as connected when their real surfaces touch (gap under about 4 mm), or when
↪  a joint joins them at a place where both parts are: the joint's origin within 2 cm of
↪  each part's surface. A joint whose origin is far from either part connects nothing.
↪  Parts that interpenetrate a lot collide. Anything not connected to the rest is
↪  floating.

DECLARATIONS (all are scored)
- role: say what each part is, in plain words ("seat", "backrest", "caster wheel", ...).
- material: choose one per part from list_materials.
- joints: declare every place the real object moves. add_joint(joint_id, type, parent,
↪  child, axis, origin, limits). type is one of fixed, revolute, continuous, prismatic,
↪  cylindrical, ball. axis is a direction in world coordinates; origin is a world point
↪  on the axis where parent and child meet;
limits = [low, high] in radians (revolute) or metres (prismatic, cylindrical = the sliding
↪  range); omit limits for continuous. The child moves relative to the parent, carrying
↪  whatever is attached to the child.
- assembly sequence: set_assembly_sequence with every part once, in the order a person
↪  would build the object, each step optionally with the direction the part travels as it
↪  is put in place (e.g. [0, 0, -1] for lowering it from above).

ROUNDS AND BUDGETS
- submit ends a round. If review rounds remain you are shown a check of the design and
↪  renders, and you may revise it and submit again. After the final round the episode
↪  ends.
- check and render are metered; use them to verify, not to search. Tool calls are
↪  budgeted.

Keep reasoning brief. Prefer making tool calls over explaining.
```

Task prompt. The user message specifies the target category, optional reference image, optional fine-grained attributes and functional constraints, and the current environment tier. For example, the main Tier-B multimodal condition for an office chair is:

Example user prompt: Tier B with reference image

```
TASK: According to image W2, build an office chair. The condition image W2 is attached.
Tier B: Use catalog parts, and create new parts where the catalog has none that fit. A
↪  picture of every catalog part, labelled with id, name and size, is attached.
↪  view_catalog_sheet shows it again. Start with list_parts and list_materials.
```

For name-only conditions, the first line is simply "Build an office chair." For image-conditioned tasks, the corresponding image identifier is prepended. Attribute-conditioned tasks append descriptions of required components, materials, and other object properties. Functional conditions additionally provide subsystem and kinematic constraints, such as specifying that an office chair seat rotates relative to its base about a vertical axis. Tier A instructs the model to use catalog parts only, while Tier C states that no catalog exists, requires every placed component to be created, and begins with `list_materials` followed by `create_part`. For models without visual capability, image-conditioned conditions are preserved rather than silently converted to text: the message explicitly states that the task refers to an image that the model cannot see. This allows the same benchmark condition to be retained while exposing the model's actual input limitation.

A.1.2 CLOSED-SOURCE AGENT HARNESS

Claude models. Claude Haiku 4.5, Sonnet 5, Opus 5, and Fable 5.1 are evaluated through the Claude command-line interface. Each episode is placed in an isolated working directory and the model is given only the session interface and the image files returned by that session. The agent first requests the complete LMBuild prompt and subsequently invokes environment tools through shell commands of the form

```
.venv_eval/bin/python -m ppbench.v2.session call --state <STATE> <TOOL_NAME>
↪  '<JSON_ARGUMENTS>'
```

Each invocation returns one JSON object. Tools that produce images return PNG paths, which Claude may inspect using its image-reading capability. The outer wrapper is:

Claude shell wrapper

```
You are the model being evaluated in a 3D assembly benchmark. Work only through the
↪  session command below; do not edit, create or delete any files, and do not run any
↪  other programs except reading image files.

Your session state file: {state}

1. Run: {cmd} prompt --state {state}
   Read the whole output: it is your task, your tools and how to call them. Open every
   ↪  image file it names with your file-reading tool.

2. Build the object with the tools. Tool calls look like:
   {cmd} call --state {state} <tool_name> '<JSON arguments>'
   Each call prints one JSON result; image results are PNG paths you open to look at.

3. The episode may have several rounds: each submit ends a round and returns a review.
↪  Keep working until a submit result says "episode_done": true. If a call returns an
↪  error, read it and correct the call.

Reply at the end with one line: the number of parts and joints in your final design.
```

GPT models. GPT-5.6 Sol and GPT-6 Astra use the same session environment but are driven through codex. Their outer wrapper is slightly more restrictive with respect to filesystem access:

GPT shell wrapper

```
You are the model being evaluated in a 3D assembly benchmark. Work only through the
↪  session command below. Do not edit, create, or delete files directly. Do not inspect
↪  the reference assembly, evaluator data, other results, or repository code. You may
↪  read only the condition image, catalog sheets, and renders named by your session.

Your session state file is {state}.

Run `{cmd} prompt --state {state}` first and read its full output. Open the image files it
↪  names. Build the object using only session tool calls such as `{cmd} call --state
↪  {state} <tool_name> '<JSON arguments>'`. Each successful submit saves a round and
↪  returns feedback. Continue revising the same design until a submit returns
↪  `episode_done: true`. If the saved session is already partly complete, call get_scene
↪  and continue from that state; never start over. Inspect your renders and correct
↪  errors before submitting. At the end, report the final part and joint counts.
```

Thus, Claude and GPT receive the same LMBuild environment prompt and tool interface; their outer wrappers differ only in provider-specific execution and isolation instructions. GPT episodes permit up to three execution attempts.

A.1.3 OPEN-SOURCE LLM/VLM AGENT HARNESS

Open-source LLMs and VLMs are served using vLLM through an OpenAI-compatible chat endpoint. This group includes Qwen3-VL, Qwen3.5, Gemma-4, InternVL3.5, MiniCPM-V, GPT-OSS, Ministral-3, ERNIE-4.5-VL, Llama-3.1-8B, etc. Unlike the shell-based agents, these models interact with LMBuild through an in-process agent loop.

The system message is constructed as

Table 3: Comparison of domain-specific structure-generation systems and LLM agents in terms of the information received and produced throughout the construction pipeline. While LLM agents interact seamlessly with LMBuild, domain-specific systems cover only a subset.

#	System	Pipeline	Input	Geometry	Parts	Roles	Joints	Materials	Sequence
1	BrickGPT	Generates LEGO Brick Structures	Text	✓	✓	✗	✗	✗	✗
2	LegoACE	Generates LEGO Brick Structures	Text	✓	✓	✗	✗	✗	✗
3	PartCrafter-15	Generates a mesh of 15 parts.	Image	✓	✓	✗	✗	✗	✗
4	PartCrafter-8	Generates a mesh of 8 parts.	Image	✓	✓	✗	✗	✗	✗
5	PartPacker	Generates part meshes.	Image	✓	✓	✗	✗	✗	✗
6	Cube3D+CubePart	Generate geometry and decompose	Text	✓	✓	✓*	✗	✗	✗
7	Particulate +(3)	Re-segment PartCrafter-15 Mesh	Image	✓	✓	✗	✓	✗	✗
8	Particulate +(4)	Re-segment PartCrafter-8 Mesh	Image	✓	✓	✗	✓	✗	✗
9	Particulate +(5)	Re-segment PartPacker Mesh	Image	✓	✓	✗	✓	✗	✗
10	Particulate +(6)	Re-segment Cube-3D Mesh	Text	✓	✓	✗	✓	✗	✗
11	PhysX-Anything	Generation aligned with Physics	Image	✓	✓	✓	✓	✓	✗
–	**LLM Agents (19)**	**Interact with LMBuild**	**Multimodal**	✓	✓	✓	✓	✓	✓

`system_prompt(tier) + tool_docs(tier) + tool-call protocol,`

while the user message contains the task text, the condition image when the model supports vision, and the catalog sheet for Tier A/B. Condition images are resized to a maximum side length of 1024 pixels, while catalog sheets are resized to a maximum side length of 1600 pixels. Because the shared vLLM deployment does not require model-specific native tool parsers, LMBuild uses the following text protocol for all locally served agents:

Open-source tool-call protocol

```
Write each tool call as a block on its own: <tool_call>{"name": "place_part", "arguments":
↪  {"part_id": "P001", "instance_id": "wheel_1", "position": [0, 0, 0.03]}}</tool_call>

You may write several blocks in one reply; they run in order and you then see
every result. Arguments are strict JSON (double quotes, no comments, no trailing commas).
```

Tool outputs are returned as user messages using the form

```
<tool_result name="TOOL_NAME">{...JSON result...}</tool_result>
```

The parser first searches for explicit `<tool_call>` blocks and then supports a small number of model-specific fallback formats, including ERNIE-style `<name>`/`<arguments>` outputs, standalone or fenced JSON, and function-like expressions. Automatic repair is deliberately limited to evaluating simple arithmetic expressions in numeric arguments and closing truncated JSON structures. When parsing fails, the model receives the concrete parsing error and is instructed to resend a strict-JSON tool call. If a model produces no environment action, it is reminded that the interaction contains no human operator and that it should continue through tools or call `submit` once the object is complete. An episode is terminated after three consecutive responses containing no executable tool call. All locally served models use temperature 0.6 and random seed 0. Model-specific serving modifications are used only when required by the underlying model implementation.

A.1.4 Domain-Specific Generation Systems

Domain-specific structure generators are not forced into the LMBuild agent interface when their native architectures do not support iterative tool use. Instead, each system receives the subset of the benchmark condition supported by its original inference pipeline and performs a single generation. We retain each method's native representation and prompting procedure rather than adding capabilities unavailable to the original system. For these methods, Tier B is operationally identical to Tier A because their native generation pipelines do not expose LMBuild's optional `create_part` action.

A.2 Environment Details for LMBuild

Implementation details of the LMBuild environment. LMBuild is a stateful, pure-Python tool environment. Entire episode state coming from LMBuild (placed instances with poses, created-part programs, joints, assembly sequence, counters and a full call trace) is treated as one JSON-serialisable dictionary, so the same episode can be driven in-process or from a shell, and it can be checkpointed after every round and replayed exactly.

World frame and parts All geometry of LMBuild lives in one metric world frame: $+Z$ up, ground plane at $z = 0$, object back facing Y. Catalog parts are triangle meshes taken from segmented reference assets and stored as glTF. On loading, trimesh converts each mesh with the defined convention and re-centers it on its bounding box. Therefore reduce to exact rigid transforms: a translation plus XYZ Euler rotations, or face-centered alignment of bounding boxes.

Part creation The part creation tool accepts a declarative CSG program. The program is a DAG of at most 96 nodes over box, cylinder, sphere, torus, extrude and revolve primitives. These can be combined with union, difference, intersection and hull, and moved with transform and mirror. The program is compiled with manifold3d (v3.5), which guarantees watertight 2-manifold output. Every created part is therefore a valid solid with an exact volume. Parameters are range-checked (0.5 mm–3 m), curved primitives are tessellated with 48 segments, and parts that fall apart into disconnected pieces are flagged.

Physical feedback All physical feedback comes from one geometry kernel, built on NumPy and SciPy, that is shared with the evaluator. Each placed part is voxelised on a 4-mm grid: area-proportional surface sampling, then scipy gives a solid occupancy grid.

Material Definitions Declared materials are restricted to 23 submaterials in 9 classes. Their densities are medians over 5,510 Artiverse annotations.

Rendering Renders are produced out-of-process by a headless Blender (bpy 5.2 LTS) worker. The posed scene is exported as a GLB, with each part in a distinct hue so that parts can be told apart. Each image is rendered with Cycles on the CPU (448 px, 16 samples with denoising, AgX view transform) under three sun lights and above a ground plane. The camera has a 50 mm lens and auto-framing from the design's bounding sphere. Six fixed (azimuth, elevation) viewpoints are available, including one matching the condition image's camera. The rendered views are tiled with PIL. The catalog contact sheet is pre-rendered once in the same way.

Running restrictions The environment never repairs a model's action. Invalid calls return explicit error messages and are counted. Budgets limitation include: 160 tool calls per round to restrict unlimited interactions. 6 check calls and 4 render calls per episode; the review returned by submit is free. An episode consists of up to three cumulative rounds (R1-R3). Each submit returns a free review (a check plus four views), and the design is checkpointed.

Comparability Across System Families LMBuild includes both general-purpose LLM agents and domain-specific structure-generation systems to provide a broad view of current capabilities for structured 3D generation. Comparisons should nevertheless be interpreted with respect to the outputs natively supported by each system. General-purpose LLM agents receive the same LMBuild task specification, environment semantics, tool definitions, and interaction budgets, and are evaluated end-to-end across geometry, parts, joints, materials, and assembly sequence. In contrast, domain-specific systems are evaluated using their native inference pipelines and may produce only a subset of these outputs, as summarized in Table 3. We therefore report a metric as N/A when the information required by that metric lies outside a system's native output space, rather than treating the absence of an unsupported output as a generation failure. Comparisons between LLM agents and domain-specific systems should consequently be interpreted primarily on their shared applicable metrics, while the complete LMBuild evaluation for LLM agents reflects their ability to solve the full end-to-end construction task. We include domain-specific systems not to imply identical interfaces or capabilities, but to contextualize how specialized generation methods compare with general-purpose agents on the portions of the structure-generation pipeline they support.

Information exposed to agents. The environment communicates generic, system-independent construction rules and diagnostic feedback needed to use its tools, while target-specific evaluation sheets

and final metric outcomes are not exposed to agents. We view these generic rules as part of the benchmark interface rather than hidden test labels. The functional-prompt experiments intentionally expose additional task requirements and should be interpreted as measuring an agent's ability to translate explicit specifications into structures, whereas the baseline also tests whether those requirements can be inferred from a category label and reference image. Future work may separately study settings with weaker feedback, hidden interface tolerances, or interactive requirement elicitation.

We provide the complete list of 19 tools implemented in the LMBuild environment that models can invoke when constructing structures, together with description of the functionality of each tool.

Detailed Description for Tool Set Available in LMBuild

Tool: list-parts(query);

Function: Lists the parts currently available to the agent, including catalog parts and any parts created by the model, together with their IDs, names, and dimensions.

Tool: inspect-part(part-id);

Function: Inspects a specified part and returns its information, including dimensions, watertightness, surface area, and thumbnail path.

Tool: view-catalog-sheet();

Function: Displays a labeled visual sheet containing all parts available in the catalog. This tool is unavailable when there is no sub-part pool.

Tool: list-materials();

Function: Lists the available construction materials, including their material classes and physical densities.

Tool: place-part(part-id, instance-id, position, rotation, role-description);

Function: Places a selected part into the 3D scene at a specified position and optional rotation, and specify its role if possible. It also returns feedback about its bounding box, contacts, collisions, and ground penetration.

Tool: set-material(part-id, instance-id);

Function: Specify the material of a selected part.

Tool: move-part(part-id, position, rotation);

Function: Moves or rotates an existing part using either an absolute target position or a relative translation, while returning updated contact, collision, and ground-penetration information.

Tool: attach(instance-id, anchor, target, target-anchor, offset);

Function: Aligns an anchor of one part with an anchor of another part or with the ground, enabling structured placement without manually specifying exact coordinates.

Tool: measure(part a, part b);

Function: Measures the spatial relationship between two parts, including their minimum surface gap, closest points, touching status, and overlap volume.

Tool: remove-part(instance-id);

Function: Removes a placed part from the current structure, together with its associated joints and its entry in the assembly sequence.

Tool: set-part-info(instance-id, role, material);

Function: Updates the information associated with a placed part, such as its functional role and material assignment.

Tool: remove-joint(joint-id);

Function: Removes an existing joint declaration between two parts.

Tool: add-joint(joint-id, type, parent, child, axis, origin);

Function: Defines a joint relationship between two parts. Supported joint types include fixed, revolute, continuous, prismatic, cylindrical, and ball joints.

Tool: set-assembly-sequence(steps);

Function: Specifies the intended assembly order of the generated structure, with an optional insertion direction for each assembly step.

Tool: create-part(name, program);

Function: Creates a new custom part from a constructive solid geometry program. This tool is available when create new parts is enabled.

Tool: get-scene();

Function: Returns the current state of the construction environment, including all placed parts, their poses, bounding boxes, roles, materials, joints, assembly sequence, and remaining interaction budgets.

Tool: check();

Function: Use rule-based implementations to check the current structure for potential issues, including collisions, disconnected components, ground contact, stability problems, joint issues, and missing declarations. This tool is metered.

Tool: render(view);

Function: Generates visual renderings of the current structure from one or more specified viewpoints, allowing the agent to inspect and refine its design. This tool is metered.

Tool: submit();

Function: Ends the current interaction round. If additional rounds remain, it returns structural feedback and multiple rendered views before advancing to the next round; after the final round, the task is completed.

A.3 Data Selection and Data Examples in LMBuild

Data Selection Strategy. LMBuild is constructed from a heterogeneous pool of approximately 367K raw 3D assets. Because these datasets differ substantially in geometry quality, semantic annotation, part decomposition, and kinematic information, we do not directly sample benchmark

objects from the combined pool. Instead, we apply a five-stage selection and grounding pipeline designed to ensure that retained objects are structurally valid, sufficiently articulated, semantically identifiable, diverse in their underlying construction, and grounded in independently retrievable real-world knowledge.

Candidate Admission and Structural Validity. We first remove assets that are unsuitable for evaluating part-based structure generation. An object is admitted into the candidate catalog only if it contains at least four separately identifiable components, its geometry can be successfully loaded and rendered, and any available joint annotations can be resolved to parts belonging to the same object. These requirements exclude corrupted meshes, effectively monolithic objects, incomplete assemblies, and assets whose kinematic annotations refer to missing or external components. After this initial filtering step, approximately 74.7K objects remain as structurally valid candidates.

Source-aware and Diversity-aware Sampling. We next construct a manageable candidate subset while preserving structural diversity within each source and semantic category. Because the source datasets differ substantially in their characteristic level of decomposition, we use source-specific part-count ranges rather than imposing a single global threshold. We retain Artiverse objects with 4–60 parts, PartNeXt objects with 5–40 parts, LEGO assemblies containing 25–150 bricks, and Fusion 360 assemblies with 6–60 components and at least one movable joint. These ranges remove both trivial instances and extremely large assemblies whose complexity is dominated by dataset-specific decomposition rather than meaningful object structure. Within each source-category pair, we then perform diversity-aware sampling instead of random selection. Each candidate object is represented by a structural descriptor containing counts of semantic part roles together with counts of moving-part and joint-type combinations. We first select the instance closest to the component-wise median of these descriptors, yielding a representative medoid-like example. Subsequent instances are chosen using farthest-point sampling under ℓ_1 distance, repeatedly selecting the object that is most dissimilar from the examples already retained. This procedure favors coverage of distinct structural configurations rather than repeatedly selecting near-duplicate instances of the same design. Dataset-specific metadata are additionally used when necessary: LEGO categories are recovered from captions and assemblies are prioritized according to movable connectors such as hinges, axles, and ball joints, whereas Fusion 360 categories are assigned from rendered views using a VLM.

Semantic Concept Grounding and Visual Verification. Structural validity alone does not guarantee that a CAD instance corresponds to a well-defined real-world object category. We therefore ground every retained category to both Wikidata and WordNet. For each category name, we retrieve the top Wikidata search candidates and the corresponding WordNet noun synsets. A language model is permitted only to select among these retrieved candidates, rather than freely generating an entity identifier. The selected Wikidata item and WordNet synset must denote the general physical object type represented by the category; entities corresponding to brands, named products, artworks, locations, or other non-category concepts are rejected. Categories for which no suitable Wikidata entity can be identified are removed. We additionally verify the category assignment at the individual-instance level. A VLM observes two rendered views of each candidate object and determines whether the asset both matches the intended category and represents a reasonably complete instance of that object. This step removes fragments, isolated subassemblies, severely incomplete models, and semantically mislabelled examples that may have passed the metadata-based filters.

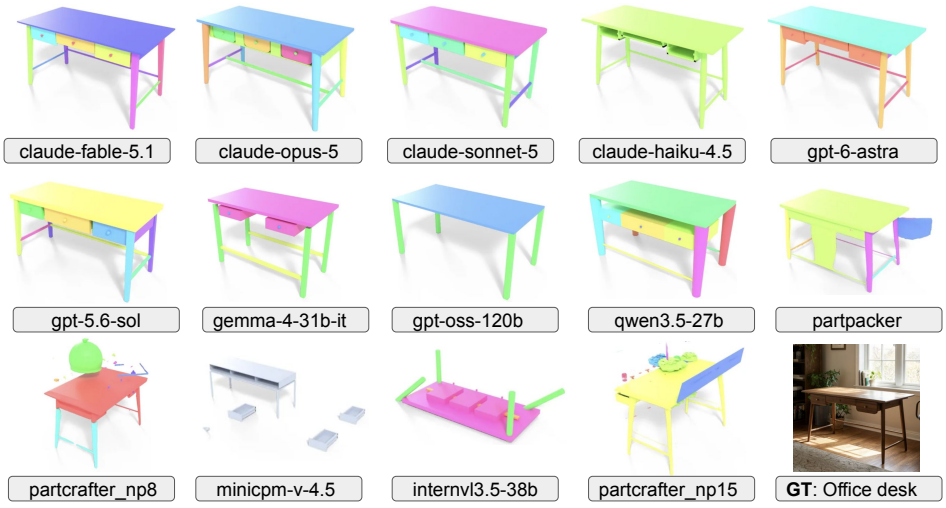
Kinematic Coverage and Core-set Construction. From the grounded candidate pool, we construct the LMBuild-Core benchmark by explicitly covering qualitatively different forms of mechanical organization. Rather than selecting the 50 core tasks solely according to object category frequency, we organize them into six broad kinematic families: **(A)** static load-bearing structures, **(B)** hinged mechanisms, **(C)** sliding mechanisms, **(D)** rolling or rotating systems, **(E)** serial kinematic chains, and **(F)** structures containing many joints attached to a common body. These families are intended to expose models to substantially different requirements for connectivity, motion, joint specification, and physical realization. The final 50 core objects span a wide range of complexity, from small assemblies containing only four parts to highly decomposed structures containing as many as 417 parts. The remaining selected data form LMBuild-Full, containing 2,349 additional objects from 152 semantic categories. LMBuild-Full may share object categories with LMBuild-

Core, but it contains no overlapping object instances, allowing the extended set to preserve broader intra-category structural variation without duplicating the tasks used in our primary evaluation.

Retrieval-based Real-world Knowledge Grounding. LMBuild further augments each selected object with external knowledge describing its expected components, materials, functional attributes, and mechanisms. Importantly, these annotations are not manually authored from the benchmark designers' prior knowledge and are not inferred directly from the CAD geometry. Instead, we construct a reproducible evidence bundle from Wikidata, Wikipedia, WordNet, and, only as a corroborating source, ConceptNet. For each category's Wikidata entity, we retrieve structured statements including has part, material, use, uses, powered by, and inverse part of relations. To recover information stated at a more general taxonomic level, we additionally traverse up to three levels of subclass of ancestors, retaining at most three ancestors per level and terminating at overly generic concepts such as physical object, product, or equipment. Part and material information may be inherited from these ancestors, but inherited parts are retained only when they have an English Wikipedia article and their own Wikidata entry independently confirms an appropriate part of relation. This conservative inheritance rule reduces the risk of propagating overly broad taxonomic properties to specific object categories. We also pin specific Wikipedia revisions for the target object, identified components, and mechanism-level concepts relevant to the object's kinematic family. For example, hinged structures may retrieve evidence from articles such as Hinge, rotating systems from Wheel or Caster, and articulated mechanisms from Kinematic pair. For identified moving parts, we additionally retrieve a relevant Wikipedia article when available. WordNet part and substance meronyms are incorporated as supplementary structured evidence and may be inherited over up to two hypernym levels. ConceptNet is used only to corroborate evidence already supported by stronger sources and is never treated as the sole basis for benchmark annotations.

Evidence-constrained Claim Extraction and Verification. The retrieved evidence is converted into benchmark annotations through a two-model extraction and verification procedure. Qwen3.5-27B first proposes candidate claims describing object parts, attributes, subsystems, and kinematic relationships. Every proposed claim must include a supporting quotation from the retrieved evidence bundle, and claims whose quoted evidence cannot be matched verbatim to the source text are automatically discarded. This prevents the extraction model from introducing unsupported world knowledge not present in the retrieved sources. A second model, Gemma-4-31B, independently evaluates each surviving claim along two dimensions: whether it is supported by the cited evidence and whether it is typical of the general object class rather than an unusual property of a particular instance. Core properties and more detailed attributes are assessed separately. When a claim fails because its supporting evidence is incomplete or poorly localized, we permit a single re-grounding step that retrieves at most three additional sentences from the same evidence bundle; no unrestricted external generation is allowed during this stage. The final annotation therefore retains only claims that can be traced to explicit external evidence and judged to describe characteristic properties of the object category. This pipeline produces the 50-task LMBuild-Core benchmark together with the 2,349-object, 152-category extended set. Each retained knowledge claim remains traceable to its source through either a Wikidata entity and statement identifier or a pinned Wikipedia page revision and supporting quotation. This provenance is important for LMBuild because the benchmark evaluates not only whether a generated structure visually resembles an object, but also whether its parts, materials, affordances, and kinematics are consistent with independently grounded knowledge about how that object is constructed and functions.

Concrete Data Examples in LMBuild (1)



Specification the evaluator checks
Verified claims from the pinned Wikipedia and Wikidata sources; 6 further extracted claims were dropped by the verifier. The small tag says how each claim is checked on a design; hover a source for its quote. Full sources and verdicts are in the panel below.

Parts 2
- P1 **work surface** — provides a flat area for reading, writing, or using equipment
- P2 **drawers** — stores items such as office supplies and papers
- Wikidata "has part" (P527): builders hardware via Q14745, drawer, fiberboard via Q14745, furniture leg via Q14748, particle board via Q14745, plank via Q14745

Attributes 6
- A1 **material** Desks are usually made of wood or metal, though glass is sometimes used.
- A2 **material** Modern mass-produced desks may be constructed from engineered wood products like fiberboard or particle board.
- A3 **property** A desk must be sturdy to support users who lean on it while working.
- A4 **use** Desks are used for academic, professional, or domestic activities such as reading, writing, or using computers.
- A5 **operating_environment** Desks are found in schools, offices, homes, and libraries.
- A6 **user_interaction** Users typically sit at a desk, either on a separate chair or a built-in chair.

Functional subsystems 1
- F1 **Storage subsystem**: drawers, compartments, pigeonholes
  Stores office supplies, papers, and documents to keep the work surface clear.

Kinematics 1
- K1 **drawer** slides relative to desk frame *prismatic* range: from fully enclosed to fully extended

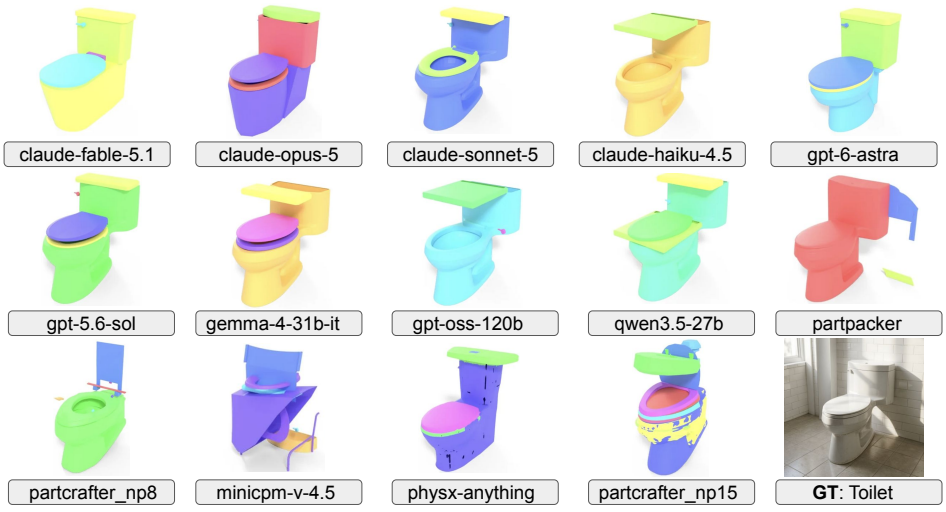
Attributes prompt

> Build a desk. Include a work surface to provide a flat area for reading, writing, or using equipment and drawers to store items such as office supplies and papers. Use wood, metal, or glass, or engineered wood products like fiberboard or particle board. Ensure the desk is sturdy to support users who lean on it while working. Design it for academic, professional, or domestic activities such as reading, writing, or using computers. Ensure it is suitable for schools, offices, homes, and libraries. Design it so users typically sit at the desk, either on a separate chair or a built-in chair.

Functional prompt

> Build a desk. Include a work surface to provide a flat area for reading, writing, or using equipment and drawers to store items such as office supplies and papers. Use wood, metal, or glass, or engineered wood products like fiberboard or particle board. Ensure the desk is sturdy to support users who lean on it while working. Design it for academic, professional, or domestic activities such as reading, writing, or using computers. Ensure it is suitable for schools, offices, homes, and libraries. Design it so users typically sit at the desk, either on a separate chair or a built-in chair. Incorporate a storage subsystem consisting of drawers, compartments, and pigeonholes to store office supplies, papers, and documents to keep the work surface clear. Install drawer slides relative to the desk frame that range from fully enclosed to fully extended.

Concrete Data Examples in LMBuild (2)



Specification the evaluator checks

Verified claims from the pinned Wikipedia and Wikidata sources; 5 further extracted claims were dropped by the verifier. The small tag says how each claim is checked on a design; hover a source for its quote. Full sources and verdicts are in the panel below.

Parts 5

- P1 **bowl** — collects human waste and holds water to seal against sewer gases
- P2 **trap** — holds water in the bowl to act as a seal against noxious sewer gases
- P3 **drain pipe** — removes effluent from the bowl to a sewer or septic tank
- P4 **flush mechanism** — releases a large volume of water rapidly to sweep waste into the drain
- P5 **cistern** — stores water for flushing in tank-style toilets
- Wikidata "has part" (P527): dry toilet, flush toilet, toilet, toilet bowl, toilet building, toilet seat, toilets in japan

Attributes 7

- A1 **material** Toilets are commonly made of ceramic (porcelain), concrete, plastic, or wood.
- A2 **use** The primary function is to collect human waste (urine and feces) and sometimes toilet paper for disposal.
- A3 **operating_environment** Flush toilets are usually connected to a sewer system in urban areas or a septic tank in isolated areas.
- A4 **power_source** Flush toilets use water as the primary medium to move waste, while dry toilets use no water.

- A5 **user_interaction** Users operate the flush by pressing a button, pushing a handle, pulling a lever, or pulling a chain.
- A6 **property** The water in the bowl acts as a barrier to prevent sewer gas from entering the building.
- A7 **use** Toilets can be designed for sitting or squatting postures depending on regional preferences.

Functional subsystems 3

- F1 **Waste Collection and Disposal**: bowl, trap, drain pipe
  Collects waste and uses a water seal to prevent gas entry while channeling waste to the sewer.
- F2 **Water Storage and Delivery**: cistern, fill valve, flush valve
  Stores water and releases it rapidly into the bowl to flush waste.
- F3 **User Interface**: flush handle, flush button, seat
  Allows the user to initiate the flush and provides a seating surface.

Kinematics 2

- K1 **flush handle** swings relative to cistern *revolute*
- K2 **flush valve (flapper)** swings relative to cistern outlet *revolute* range: lifts clear of the seat to allow flow

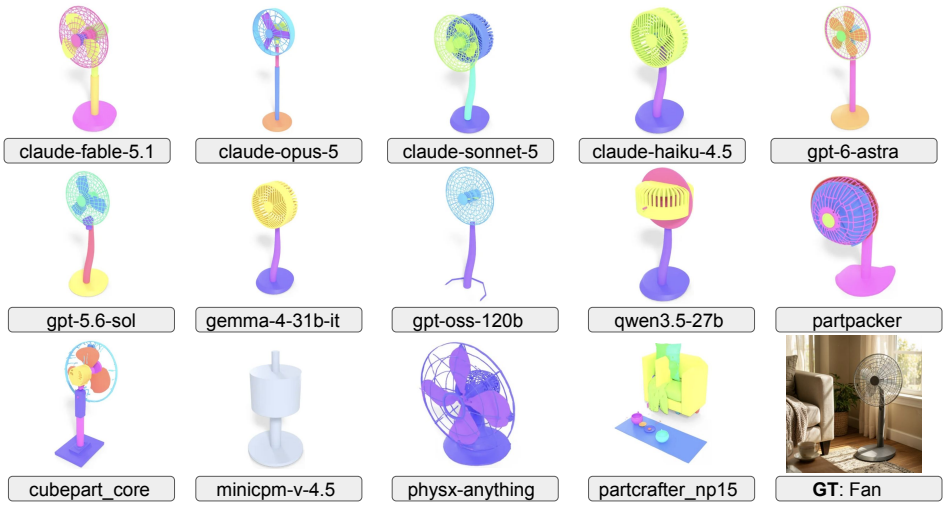
Attributes prompt

> Build a toilet. Use a bowl to collect human waste and hold water to seal against sewer gases. Include a trap to hold water in the bowl as a seal against noxious sewer gases. Attach a drain pipe to remove effluent from the bowl to a sewer or septic tank. Incorporate a flush mechanism to release a large volume of water rapidly to sweep waste into the drain. Add a cistern to store water for flushing in tank-style toilets. Construct it from ceramic, concrete, plastic, or wood. Design it to collect human waste and sometimes toilet paper for disposal. Ensure it can connect to a sewer system in urban areas or a septic tank in isolated areas. Use water as the primary medium to move waste. Allow users to operate the flush by pressing a button, pushing a handle, pulling a lever, or pulling a chain. Ensure the water in the bowl acts as a barrier to prevent sewer gas from entering the building. Design it for sitting or squatting postures.

Functional prompt

> Build a toilet. Use a bowl to collect human waste and hold water to seal against sewer gases. Include a trap to hold water in the bowl as a seal against noxious sewer gases. Attach a drain pipe to remove effluent from the bowl to a sewer or septic tank. Incorporate a flush mechanism to release a large volume of water rapidly to sweep waste into the drain. Add a cistern to store water for flushing in tank-style toilets. Construct it from ceramic, concrete, plastic, or wood. Design it to collect human waste and sometimes toilet paper for disposal. Ensure it can connect to a sewer system in urban areas or a septic tank in isolated areas. Use water as the primary medium to move waste. Allow users to operate the flush by pressing a button, pushing a handle, pulling a lever, or pulling a chain. Ensure the water in the bowl acts as a barrier to prevent sewer gas from entering the building. Design it for sitting or squatting postures. Create a Waste Collection and Disposal subsystem where the bowl connects to the trap and the trap connects to the drain pipe to channel waste and prevent gas entry. Create a Water Storage and Delivery subsystem where a fill valve carries fluid to the cistern, the cistern carries fluid to the flush valve, and the flush valve carries fluid to the bowl. Create a User Interface subsystem where a flush handle or flush button carries motion to the flush valve. Ensure the flush handle swings relative to the cistern. Ensure the flush valve swings relative to the cistern outlet and lifts clear of the seat to allow flow.

Concrete Data Examples in LMBuild (3)



Specification the evaluator checks
Verified claims from the pinned Wikipedia and Wikidata sources; 3 further extracted claims were dropped by the verifier.

Parts 7
- P1 **blades** — to act on the air and create airflow
- P2 **electric motor** — to convert electrical energy into mechanical rotation
- P3 **hub** — to connect the blades to the rotating shaft
- P4 **housing** — to direct airflow and prevent contact with blades
- P5 **blade guard** — to increase safety by preventing objects from contacting the fan blades
- P6 **oscillator gearbox** — to motion the fan from side to side
- P7 **base** — to support the fan and provide stability
- Wikidata "has part" (P527): rotor, impeller, electric motor, electrical load via Q2425052

Attributes 4
- A1 **material** Blades are generally made of wood, plastic, or metal.
- A2 **power_source** Most fans are powered by electric motors using AC or DC electricity.
- A3 **operating_environment** Fans are effective for cooling people but do not lower air temperature directly.
- A4 **property** Fans produce air flows with high volume and low pressure.

Functional subsystems 3
- F1 **Airflow generation**: blades, hub, electric motor
  Rotates blades to move air parallel to the shaft in axial fans or radially in centrifugal fans.
- F2 **Oscillation mechanism**: oscillator gearbox, oscillator shaft, base
  Moves the fan head from side to side to distribute airflow.
- F3 **Safety enclosure**: housing, blade guard
  Prevents objects from contacting the fan blades while allowing air to pass through.

Kinematics 1
- K1 **fan head** swings relative to base *revolute* range: side to side

Attributes prompt

> Build an electric fan. Use blades to act on the air and create airflow, an electric motor to convert electrical energy into mechanical rotation, and a hub to connect the blades to the rotating shaft. Include a housing to direct airflow and prevent contact with blades, a blade guard to increase safety by preventing objects from contacting the fan blades, an oscillator gearbox to motion the fan from side to side, and a base to support the fan and provide stability. The blades are generally

made of wood, plastic, or metal. Most fans are powered by electric motors using AC or DC electricity. Fans are effective for cooling people but do not lower air temperature directly. Fans produce air flows with high volume and low pressure.

Functional prompt

Build an electric fan. Use blades to act on the air and create airflow, an electric motor to convert electrical energy into mechanical rotation, and a hub to connect the blades to the rotating shaft. Include a housing to direct airflow and prevent contact with blades, a blade guard to increase safety by preventing objects from contacting the fan blades, an oscillator gearbox to motion the fan from side to side, and a base to support the fan and provide stability. The blades are generally made of wood, plastic, or metal. Most fans are powered by electric motors using AC or DC electricity. Fans are effective for cooling people but do not lower air temperature directly. Fans produce air flows with high volume and low pressure. For airflow generation, rotate blades to move air using the electric motor and hub, where the hub to blades carries motion. For the oscillation mechanism, use the oscillator gearbox and base to move the fan head from side to side to distribute airflow, where the oscillator gearbox to oscillator shaft carries motion and the oscillator shaft to base carries motion. For the safety enclosure, use the housing and blade guard to prevent objects from contacting the fan blades while allowing air to pass through, where the blade guard to motor case carries load. Ensure the fan head swings relative to base with a side to side range.

Concrete Data Examples in LMBuild (4)



Specification the evaluator checks

Verified claims from the pinned Wikipedia and Wikidata sources; 3 further extracted claims were dropped by the verifier. The small tag says how each claim is checked on a design; hover a source for its quote. Full sources and verdicts are in the panel below.

Parts 5

- P1 **heating chamber** — encloses the space where materials are exposed to heat
- P2 **heating element** — generates heat to raise the temperature inside the chamber
- P3 **rack** — supports food items inside the chamber for even heating
- P4 **thermostat** — regulates the temperature by turning the heating source on or off
- P5 **control interface** — allows the user to set temperature and operating modes
- Wikidata "has part" (P527): oven, grate, zug

Attributes 6

- A1 **use** Ovens are used to expose materials to a hot environment for cooking, baking, or industrial processing.
- A2 **power_source** Ovens are typically powered by electricity, natural gas, or combustion of solid fuels like wood or coal.
- A3 **property** The interior of an oven reaches high temperatures to cook or process materials.
- A4 **operating_environment** Ovens are commonly found in kitchens, restaurants, and industrial facilities.
- A5 **user_interaction** Users interact with ovens via controls to set temperature, time, and cooking modes.
- A6 **material** Oven chambers are constructed from fireproof materials such as brick, stone, clay, or metal.

Functional subsystems 4

- F1 **Heating System**: heating element, fuel source, combustion chamber
  Generates and distributes heat within the chamber to cook or process materials.
- F2 **Temperature Control System**: thermostat, control interface, sensors
  Maintains the chamber temperature at a user-defined setpoint by regulating the heating source.
- F3 **Access System**: door, hinge, handle
  Provides a sealable opening for loading and unloading materials while retaining heat.
- F4 **Support System**: rack, shelf
  Holds food or materials at specific heights within the chamber for optimal heat exposure.

Kinematics 2

- K1 **door** swings relative to oven frame *revolute*
- K2 **rotisserie spit** rotates relative to oven interior *revolute*

Attributes prompt

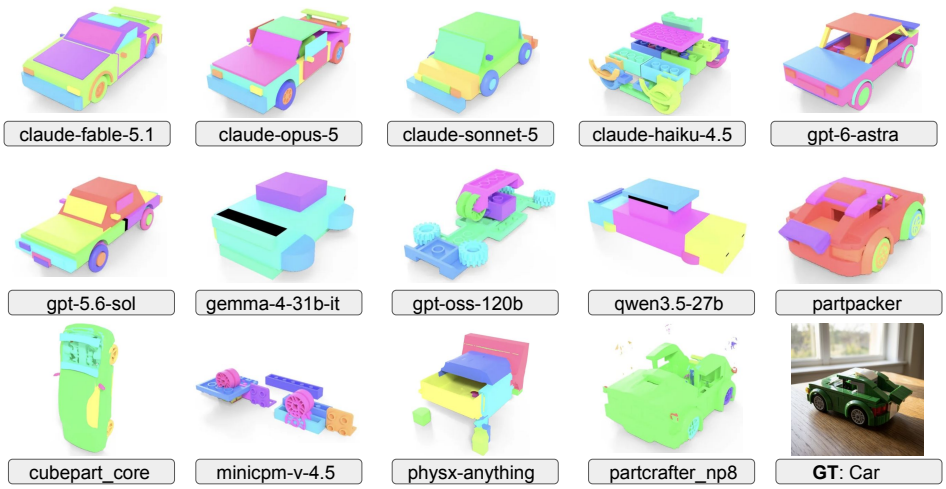
> Build an oven. Use it to expose materials to a hot environment for cooking, baking, or industrial processing. Power it with electricity, natural gas, or combustion of solid fuels like wood or coal. Ensure the interior reaches high temperatures to cook or process materials. Place it in kitchens, restaurants, or industrial facilities. Allow users to interact via controls to set temperature, time, and cooking modes. Construct the heating chamber from fireproof materials such as brick, stone, clay, or metal. Include a heating chamber to enclose the space where materials are exposed to heat. Include a heating element to generate heat to raise the temperature inside the chamber. Include a rack to support food items inside the chamber for even heating. Include a thermostat to regulate the temperature by turning the heating source on or off. Include a control interface to allow the user to set temperature and operating modes.

Functional prompt

> Build an oven. Use it to expose materials to a hot environment for cooking, baking, or industrial processing. Power it with electricity, natural gas, or combustion of solid fuels like wood or coal. Ensure the interior reaches high temperatures to cook or process materials. Place it in kitchens, restaurants, or industrial facilities. Allow users to interact via controls to set temperature, time, and cooking modes. Construct the heating chamber from fireproof materials such as brick, stone, clay, or metal. Include a heating chamber to enclose the space where materials are exposed to heat. Include a heating element to generate heat to raise the temperature inside the chamber. Include a rack to support food items inside the chamber for even heating. Include a thermostat to regulate the temperature by turning the heating source on or off. Include a control interface to allow the user to set temperature and operating modes. Implement a Heating System using a heating element, fuel source, and combustion chamber to generate and distribute heat within the chamber. Implement a Temperature Control System using a thermostat, control interface, and sensors to maintain the chamber temperature at a user-defined setpoint. Implement an Access System using a door, hinge, and handle to provide a sealable opening, where the door to the heating chamber carries load. Implement a Support System using a rack and shelf to hold materials, where the rack to the

heating chamber carries load. Ensure the door swings relative to the oven frame. Ensure the rotisserie spit rotates relative to the oven interior.

Concrete Data Examples in LMBuild (5)



Specification the evaluator checks
Verified claims from the pinned Wikipedia and Wikidata sources; 11 further extracted claims were dropped by the verifier. The small tag says how each claim is checked on a design; hover a source for its quote. Full sources and verdicts are in the panel below.

Parts 9
- P1 **Engine** — converts fuel energy into mechanical motion to propel the vehicle
- P2 **Wheels and Tires** — support the vehicle load and provide traction on the road surface
- P3 **Steering System** — allows the driver to control the direction of the vehicle
- P4 **Brakes** — inhibit motion to slow or stop the vehicle
- P5 **Transmission** — transfers engine power to the wheels and adjusts torque and speed
- P6 **Fuel Tank** — stores the flammable fluid required for the engine to operate
- P7 **Seats** — provide a place for occupants to sit while the vehicle is in motion
- P8 **Doors** — allow ingress and egress for passengers and secure the vehicle interior
- P9 **Suspension** — absorbs shock impulses from the road to maintain tire contact and ride comfort

Attributes 6
- A1 **power_source** Most cars are powered by internal combustion engines burning fossil fuels, though electric motors are increasingly common.
- A2 **material** Car bodies are typically constructed from steel, aluminum, or increasingly from composite materials like carbon fiber.
- A3 **use** Cars are primarily used to transport people rather than cargo on roads.
- A4 **operating_environment** Cars are designed to operate primarily on paved roads.
- A5 **property** Modern cars typically weigh between 1 and 3 tonnes.
- A6 **user_interaction** Drivers control the vehicle using a steering wheel, pedals for braking and acceleration, and a gear shift.

Functional subsystems 2
- F1 **Propulsion System**: Engine, Fuel Tank, Transmission, Clutch, Drive Shaft, Differential, Axle
  Generates power from fuel and transmits it to the wheels to move the vehicle.

- F2 **Steering System**: Steering Wheel, Steering Column, Rack and Pinion, Tie Rods, Steering Knuckle, Ball Joint
  Translates driver input into the turning of the front wheels to change direction.

Kinematics 5

- K1 **Steering Wheel** rotates relative to Steering Column *revolute*
- K2 **Front Wheels** swings relative to Vehicle Chassis *revolute*
- K3 **Car Door** swings relative to Vehicle Body *revolute*
- K4 **Side Windows** slides relative to Door Frame *prismatic* axis: vertical axis
- K5 **Brake Pedal** swings relative to Dashboard/Floor *revolute*

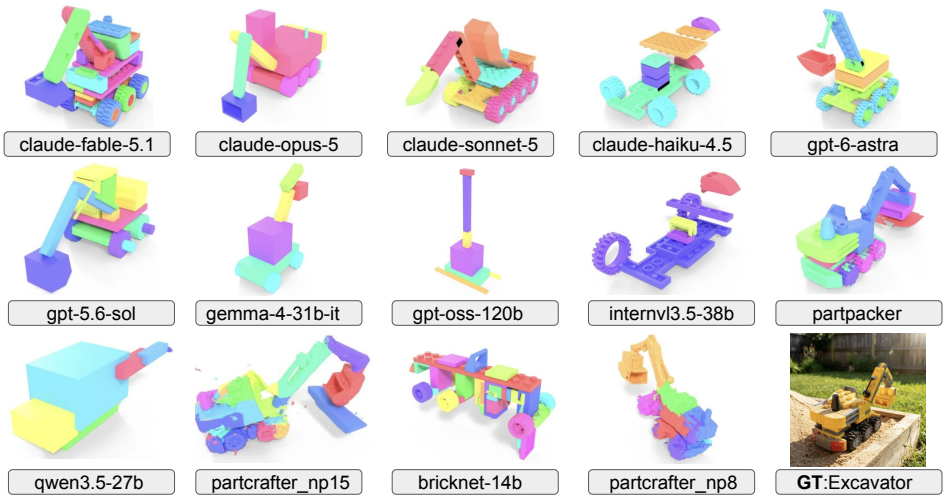
Attributes prompt

> Build a car. Use an engine, wheels and tires, a steering system, brakes, a transmission, a fuel tank, seats, doors, and suspension. The vehicle is powered by internal combustion engines or electric motors and constructed from steel, aluminum, or composite materials. It is primarily used to transport people on roads, specifically paved roads. The car typically weighs between 1 and 3 tonnes. Drivers interact with it using a steering wheel, braking and acceleration pedals, and a gear shift.

Functional prompt

> Build a car. Use an engine, wheels and tires, a steering system, brakes, a transmission, a fuel tank, seats, doors, and suspension. The vehicle is powered by internal combustion engines or electric motors and constructed from steel, aluminum, or composite materials. It is primarily used to transport people on roads, specifically paved roads. The car typically weighs between 1 and 3 tonnes. Drivers interact with it using a steering wheel, braking and acceleration pedals, and a gear shift. Integrate a propulsion system where the engine to transmission carries power and the differential to axle carries motion. Include a steering system where the steering wheel to steering column carries motion, the steering column to rack and pinion carries motion, the rack and pinion to tie rods carries motion, the tie rods to steering knuckle carries force, and the steering knuckle to ball joint carries force. Ensure the steering wheel rotates relative to the steering column, front wheels swing relative to the vehicle chassis, the car door swings relative to the vehicle body, side windows slide vertically relative to the door frame, and the brake pedal swings relative to the dashboard or floor.

Concrete Data Examples in LMBuild (6)



Specification the evaluator checks

Verified claims from the pinned Wikipedia and Wikidata sources; 5 further extracted claims were dropped by the verifier. The small tag says how each claim is checked on a design; hover a source for its quote. Full sources and verdicts are in the panel below.

Parts 9

- P1 **Undercarriage** — Provides the mobile base with tracks or wheels for locomotion
- P2 **House** — Contains the engine, operator cab, and hydraulic systems on a rotating platform
- P3 **Boom** — Provides the primary up-and-down movement for the digging arm
- P4 **Stick (Dipper)** — Provides the digging movement to pull the bucket through the ground
- P5 **Bucket** — Serves as the container for scooping and holding excavated material
- P6 **Engine** — Converts fuel energy into mechanical power to drive hydraulic pumps
- P7 **Hydraulic Pumps** — Convert mechanical engine power into high-pressure fluid flow for actuators
- P8 **Hydraulic Cylinders** — Convert fluid pressure into linear force to move the boom, stick, and bucket
- P9 **Swing Motor** — Rotates the house 360 degrees relative to the undercarriage

Attributes 5

- A1 **power_source** Excavators typically use an internal combustion engine to drive hydraulic pumps.
- A2 **use** Excavators are used for digging trenches, holes, foundations, and material handling.
- A3 **operating_environment** Excavators operate in construction sites, mining operations, and confined spaces.
- A4 **property** Hydraulic systems in excavators operate at high pressures up to 5000 psi for main functions.
- A5 **user_interaction** Operators control the machine using joysticks that distribute digging functions across two x-y controls.

Functional subsystems 3

- F1 **Hydraulic Power System**: Engine, Hydraulic Pumps, Hydraulic Oil Tanks, Hydraulic Cylinders, Hydraulic Motors
  Converts engine mechanical energy into hydraulic fluid pressure to actuate all moving parts.
- F2 **Arm Assembly**: Boom, Stick, Bucket, Hydraulic Cylinders
  Performs the digging, lifting, and dumping motions required for excavation.
- F3 **Travel System**: Undercarriage, Tracks, Track Motors, Final Drives
  Propels the machine forward, backward, and turns via independent track drive.

Kinematics 4

- K1 **House** rotates relative to Undercarriage *revolute* axis: vertical center pin range: 360 degrees
- K2 **Stick** swings relative to Boom *revolute* range: closer and further
- K3 **Bucket** swings relative to Stick *revolute*
- K4 **Piston Rod** slides relative to Cylinder Barrel *prismatic* range: extension and retraction

Attributes prompt

> Build an excavator. Use an undercarriage for the mobile base. Build a house containing the engine, operator cab, and hydraulic systems on a rotating platform. Attach a boom for primary up-and-down movement, a stick for digging movement, and a bucket for scooping material. Include an engine to convert fuel energy into mechanical power, hydraulic pumps to convert that power into high-pressure fluid flow, and hydraulic cylinders to convert fluid pressure into linear force. Add a swing motor to rotate the house 360 degrees relative to the undercarriage. The excavator uses an internal combustion engine to drive hydraulic pumps and is used for digging trenches, holes, foundations, and material handling. It operates in construction sites, mining operations, and confined spaces. The hydraulic systems operate at high pressures up to 5000 psi. Operators control the machine using joysticks.

Functional prompt

> Build an excavator. Use an undercarriage for the mobile base. Build a house containing the engine, operator cab, and hydraulic systems on a rotating platform. Attach a boom for primary up-and-down movement, a stick for digging movement, and a bucket for scooping material. Include an engine to convert fuel energy into mechanical power, hydraulic pumps to convert that power into high-pressure fluid flow, and hydraulic cylinders to convert fluid pressure into linear force. Add a swing motor to rotate the house 360 degrees relative to the undercarriage. The excavator uses an internal combustion engine to drive hydraulic pumps and is used for digging trenches, holes, foundations, and material handling. It operates in construction sites, mining operations, and confined spaces. The hydraulic systems operate at high pressures up to 5000 psi. Operators control the machine using joysticks. Create a Hydraulic Power System using the engine, hydraulic pumps, hydraulic oil tanks, hydraulic cylinders, and hydraulic motors to convert mechanical energy into fluid pressure, where the engine connects to the hydraulic pumps. Assemble an Arm Assembly using the boom, stick, bucket, and hydraulic cylinders, where the boom carries load to the stick and the stick carries load to the bucket. Build a Travel System using the undercarriage, tracks, track motors, and final drives, where track motors and final drives carry motion to the tracks. Ensure the house rotates 360 degrees relative to the undercarriage on a vertical center pin. The stick must swing relative to the boom, and the bucket must swing relative to the stick. The piston rod must slide relative to the cylinder barrel.

Examples of Declared Joint Motion Figures 5–12 visualize joints declared in generated structures. Each row shows three poses of a generated structure under a fixed camera, illustrating the motion associated with one declared joint. Outlines identify the parts that move with the joint, and the orange rod indicates its declared axis. The moving set is determined by traversing the declared joint graph from the child part without passing through the parent part.

A.4 Detailed Evaluation Metrics for LMBuild

Unless otherwise specified, all metrics are reported on a normalized scale of $[0, 100]$. We use $\mathrm{clip}(x) = \min(1, \max(0, x))$ throughout the definitions below. Our evaluation framework comprises four hierarchical dimensions that assess the **Structural Soundness**, **Functional Affordance**, **Design Quality**, and **Physical Realization** of the generated structure. A concise overview of the evaluation question addressed by each metric is provided in Table 4.

Evaluation Framework Positioning. Most prior evaluation protocols for generated structures focus primarily on semantic alignment or geometric consistency. In contrast, our framework evaluates buildable structure generation across complementary dimensions of structural soundness, functional affordance, design quality, and physical realization. Importantly, our benchmark spans heterogeneous geometry domains, including LEGO assemblies, mechanical structures, everyday objects, and freely generated meshes, whose representations, part semantics, and physical constraints differ substantially. This cross-domain setting makes many existing domain- or representation-specific metrics difficult to apply uniformly, motivating evaluation criteria that can generalize across structure types. Several metrics necessarily introduce heuristic thresholds, tolerances, discretization choices, or weighting coefficients. These hyperparameters were selected based on geometric and physical reasoning, empirical inspection of representative structures, and practical considerations of the evaluation setting. AI tools were additionally consulted during metric development to help examine candidate formulations, identify potential edge cases, and improve the clarity of the evaluation rules; they were not used to determine benchmark outcomes or tune parameters to individual models or results. The resulting hyperparameters should therefore be interpreted as standardized evaluation parameters rather than theoretically optimal values or exact models of real-world physical behavior. All hyperparameters are fixed before benchmark-wide evaluation and applied identically across systems and outputs. We therefore view the proposed metrics as practical, interpretable, and reproducible comparative proxies rather than definitive solutions. Our goal is to provide a consistent basis for evaluating heterogeneous generated structures while making the assumptions and limitations of each metric explicit. Several dimensions, particularly design quality, human-use geometry, assembly feasibility, and operability, remain difficult to quantify in a fully unbiased and model-independent



Figure 5: Declared joint motion in office desk examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.



Figure 6: Declared joint motion in toilet examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

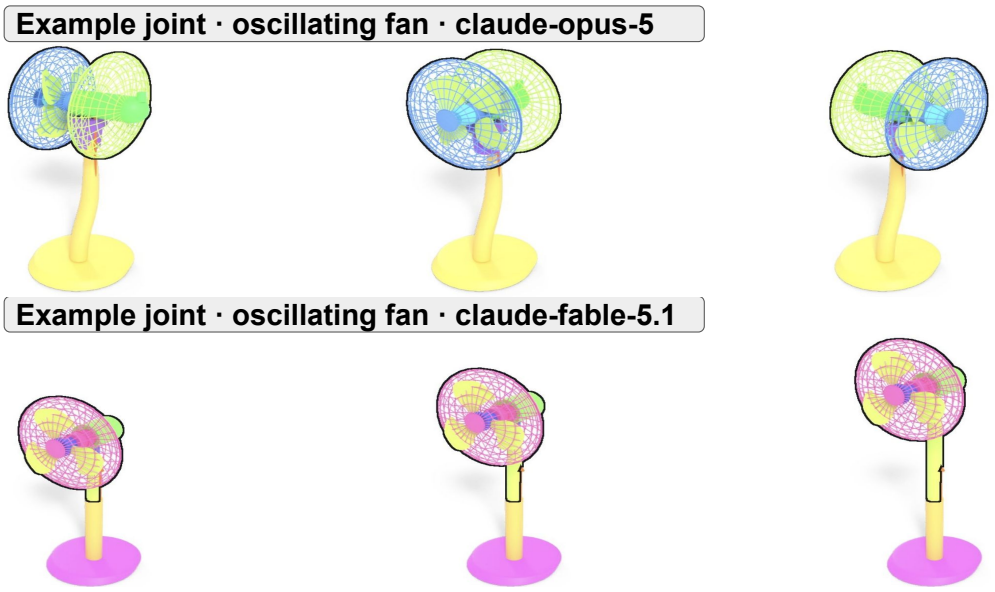


Figure 7: Declared joint motion in oscillating fan examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

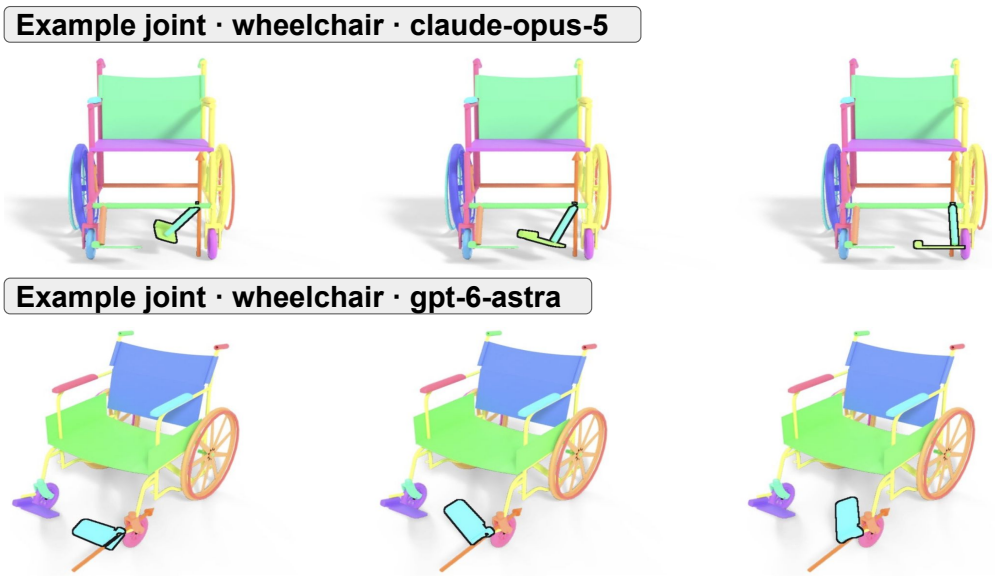


Figure 8: Declared joint motion in wheelchair examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

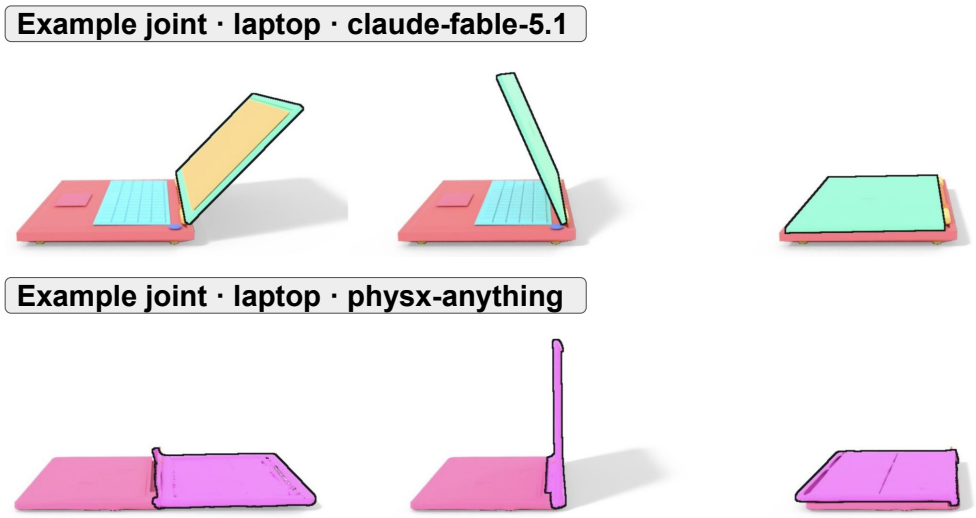


Figure 9: Declared joint motion in laptop examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

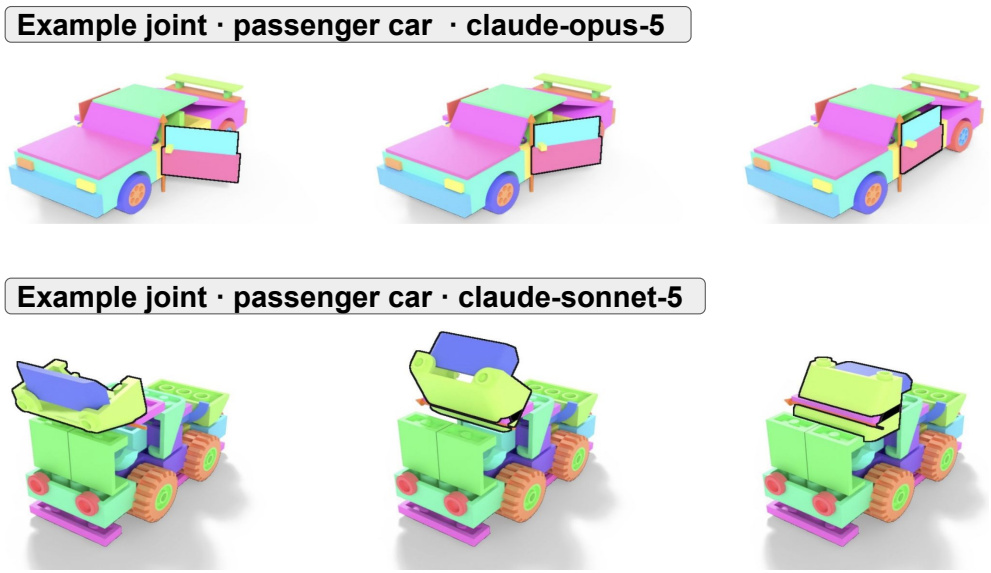


Figure 10: Declared joint motion in passenger car examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

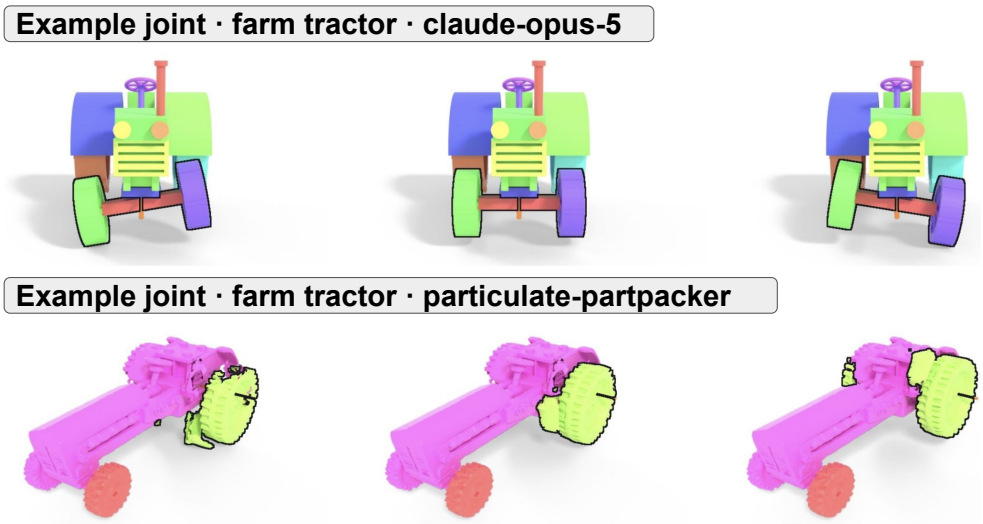


Figure 11: Declared joint motion in farm tractor examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

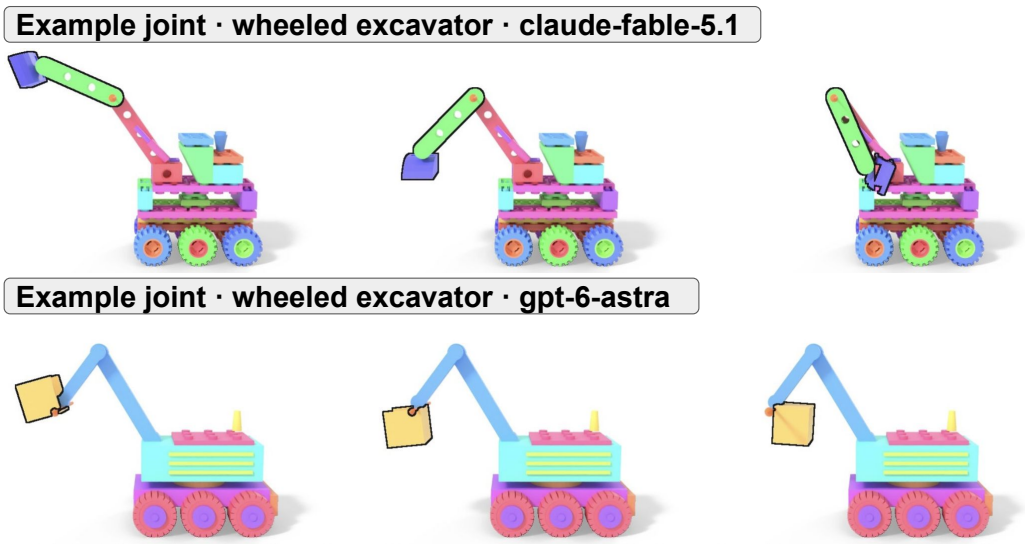


Figure 12: Declared joint motion in wheeled excavator examples. Each row shows a generated structure in three poses under a fixed camera. Outlined parts belong to the moving set, and the orange rod marks the declared joint axis.

Table 4: Illustrative questions used to interpret the LMBuild evaluation metrics, using the construction of an off-road vehicle as a running example.

Metric	Question to Ask for Evaluation
S.1 Connectivity	Are the wheels, axles, chassis, and other connected components properly attached rather than floating, detached, or disconnected?
S.2 Collision	Do the wheels and surrounding components avoid unintended interpenetration with suspension, axles, or neighboring parts?
S.3 Stability	Can the vehicle remain stably supported on the ground without tipping, collapsing, or losing appropriate contact through its wheels?
A.1 Geometry	Are the wheels appropriately shaped, sized, oriented, and positioned so that they can contact the ground and support effective driving?
A.2 Parts	Does the vehicle contain the functional components required for driving, such as wheels, axles, steering elements, and appropriate supports?
A.3 Kinematics	Can the wheels rotate around their axles and, where required, steer or articulate along the intended motion axes?
D.1 Decomposition	Is the vehicle divided into meaningful granular components rather than overly monolithic or unnecessarily fragmented parts?
D.2 Aesthetics	Does the vehicle exhibit aesthetically elegant proportions, coherent overall form, and sufficiently refined geometry as a standalone design?
D.3 Alignment	Are the wheels, cabin, and other major components arranged consistently with the intended off-road vehicle design or reference?
R.1 Sequence	Can the components be assembled in a feasible order without becoming unstable during assembly or blocking subsequent assembly steps?
R.2 Material	Are the materials appropriate for each component's physical role, such as compliant tires and sufficiently rigid and durable structural parts?
R.3 Operability	Once assembled, can the vehicle actually roll and steer through its intended mechanisms, with the required components usable?

manner. We hope this framework serves as a useful baseline and motivates further work toward more robust, generalizable, and standardized evaluation protocols for buildable structure generation.

Shared Evaluation Conventions. Each generated design consists of a collection of triangle-mesh parts represented in a coordinate system, with the vertical axis aligned to z. A design is required to additionally provide semantic role for each part, materials, declared joints with their type, axis, origin, and an assembly sequence. Evaluation uses a task-specific reference object together with curated factual claims describing required parts, functional subsystems, kinematics, and attributes extracted from **Wikipedia** knowledge. Unless otherwise specified, geometric tests use a common voxel representation with pitch $r = 4$ mm. For a part i, let G_i denote the set of voxels occupied by the part. We define $G_i^{\ominus}$ as its eroded core, obtained by removing the outermost one-voxel-thick boundary of G_i; equivalently, a voxel remains in $G_i^{\ominus}$ only when its required neighboring voxels also belong to G_i. We define $G_i^{\oplus}$ as its one-cell halo, obtained by expanding G_i outward by one voxel, such that $G_i^{\oplus}$ contains G_i together with its immediately neighboring voxels. Using this representation, two parts form an interfacing pair when

$$\mathcal{I} = \left\{(i,k) : G_i^{\oplus} \cap G_k \neq \varnothing\right\}.$$

A declared joint between two incident parts is considered **origin-valid** when its declared origin lies within 2 cm of the surface of each part. This tolerance accommodates small discrepancies introduced by voxelization, mesh discretization, and coordinate quantization while excluding joint declarations that are spatially inconsistent with the corresponding part interface. We measure proximity to the part surfaces rather than their centroids because physical joints are expected to occur at or near

the interface between connected components. We distinguish physical connectivity from mere geometric contact. Two parts are considered **physically connected** if at least one of three conditions holds: (i) their geometry forms a valid interlocking relation that mechanically constrains separation; (ii) they are connected by a geometrically valid declared fastening relation, such as a hinge, pin, screw, or other supported joint type; or (iii) for brick-based structures, they form a valid stud–receptacle connection. This hybrid definition allows the evaluation to accommodate heterogeneous construction paradigms without treating simple surface contact or proximity as sufficient evidence of attachment. These definitions serve as shared geometric primitives and are reused consistently across the structural, decomposition, assembly, and operability evaluations described below.

All per-design metric scores lies in $[0, 1]$ and will be re-sacled to $[0, 100]$. Main-table values are reported as the mean score over benchmark tasks. Models which provide empty or failed generations receive a score of zero.

Structural Soundness – S.1 Connectivity

Although the target structures in our benchmark are decomposable into multiple constituent parts, the resulting generation should ultimately represent an integrated assembled artifact rather than a collection of independently plausible but disconnected components. Connectivity therefore measures the extent to which the generated parts are physically integrated into a coherent assembly. We first construct an assembly graph in which each generated part corresponds to a node and an edge is introduced only when the relationship between two parts is supported by explicit physical evidence. Such evidence may include a valid declared joint located at a physically plausible contact region, a geometric interlock, a brick–stud connection, or another direct part-to-part attachment supported by the generated geometry. This graph-based representation allows us to distinguish nominally related parts from components that are actually supported by a plausible physical connection. Let n denote the number of generated parts, m the number of connected components in the resulting assembly graph, C_1 the largest connected component, $V(C_1)$ the total volume of the parts contained in C_1, V the total volume of the generated object, and n_{fast} the number of physically supported connections identified in the assembly. We compute

$$S_{1.1} = \frac{1}{3}\left(\frac{V(C_1)}{V} + \left(1 - \frac{m-1}{n-1}\right) + \min\left(1, \frac{n_{\text{fast}}}{n-1}\right)\right).$$

The three terms capture complementary aspects of assembly connectivity. The first term, $V(C_1)/V$, measures the fraction of the total object volume contained in the largest connected component. This volume-weighted formulation captures whether the dominant physical mass of the generated object belongs to a common assembly, while preventing a small detached decorative or auxiliary component from disproportionately reducing the score. The second term, $1 - (m-1)/(n-1)$, measures the degree of fragmentation at the part level. A fully connected assembly has $m = 1$ and therefore receives the maximum value, while the score decreases as the generated object is divided into an increasing number of disconnected components. In contrast to the volume-based term, this component remains sensitive to the number of disconnected groups even when those groups contain relatively small parts. The final term, $\min(1, n_{\text{fast}}/(n-1))$, measures whether the generated structure contains a sufficient number of connections that are explicitly supported by physical evidence. A connected graph of n nodes requires at least $n-1$ edges; we therefore use $n-1$ as a natural normalization factor. The value is capped at one because additional redundant connections may improve structural redundancy but should not increase a metric whose primary purpose is to assess whether the parts are sufficiently connected to constitute a single assembly. These three signals intentionally address different failure modes. A structure may contain one large connected body while leaving several smaller parts detached, producing a high largest-component score but a lower fragmentation score. Conversely, a structure may appear graph-connected through numerous declared relationships while those relationships are not supported by plausible physical contacts, which is reflected by the physically supported connection term. Their combination therefore evaluates not only whether most of the generated object belongs to a common component, but also whether the assembly is globally integrated and whether its connectivity is grounded in explicit physical evidence. For a valid single-part structure ($n = 1$), no inter-part connection is required; we therefore define the connectivity score as one.

Structural Soundness – S.2 Collision

A physically valid assembly should allow its constituent parts to occupy compatible regions of space without substantial unintended interpenetration. Collision therefore measures whether the generated components can coexist geometrically in a plausible assembled configuration. We do not require meshes to be strictly intersection-free, since small overlaps may arise from mesh discretization, floating-point imprecision, approximate geometric representations, or intentional connection geometry. Instead, we identify only overlaps that are sufficiently large to indicate a likely structural conflict.

For each geometrically relevant pair of parts $(i, k) \in \mathcal{I}$, let O_{ik} denote their overlap volume and V_i and V_k their respective part volumes. We classify the pair as an invalid collision only when both an absolute and a relative overlap criterion are satisfied:

$$O_{ik} > 1\,\mathrm{cm}^3 \qquad \text{and} \qquad \frac{O_{ik}}{\min(V_i, V_k)} > \tau_{ik},$$

where

$$\tau_{ik} = \begin{cases} 0.25, & \text{if a valid joint is declared between parts } i \text{ and } k, \\ 0.10, & \text{otherwise.} \end{cases}$$

The collision score is then defined as

$$S_{1.2} = 1 - \frac{\left|\left\{(i,k) \in \mathcal{I} : O_{ik} > 1\,\mathrm{cm}^3 \wedge \frac{O_{ik}}{\min(V_i,V_k)} > \tau_{ik}\right\}\right|}{|\mathcal{I}|}.$$

The two-stage criterion avoids over-penalizing harmless geometric artifacts. The absolute threshold filters negligible intersections caused by numerical tolerance or mesh approximation, while the relative threshold identifies overlaps that occupy a meaningful fraction of either participating part. A more permissive threshold is used for explicitly jointed parts, since valid connection geometries may naturally introduce localized overlap. This formulation distinguishes plausible attachment geometry from substantial unintended interpenetration, such as components occupying the same physical region or overlapping far beyond what is necessary for a connection. The metric therefore measures the proportion of geometrically plausible part interactions rather than enforcing a strict zero-intersection requirement. When no geometrically relevant part pair exists, we define the collision score as one.

Structural Soundness – S.3 Reliability

A connected and collision-free structure may still be physically implausible if it cannot maintain a stable configuration under small external perturbations. Reliability therefore evaluates whether the generated artifact can remain upright under quasi-static loading. In this work, we focus specifically on **tipping stability**, which can be estimated from the relationship between the object's center of mass and its support region without requiring a full rigid-body dynamic simulation. Let d denote the horizontal stability margin between the projection of the object's center of mass and the nearest boundary of its support footprint, h_c the height of the center of mass above the supporting surface, and h_{top} the total height of the generated structure. A larger support margin relative to the object's height generally indicates greater resistance to tipping.

We derive two complementary stability proxies. The first is the critical tilt angle,

$$\theta = \arctan\left(\frac{d}{h_c}\right),$$

which approximates the angular perturbation required for the projected center of mass to cross the boundary of the support region. A larger θ therefore indicates that the object can tolerate greater angular displacement before becoming unstable. The second proxy is

$$f = \frac{d}{h_{\mathrm{top}}},$$

which corresponds to the horizontal force applied at the top of the structure, expressed as a fraction of the object's weight, required to initiate tipping under quasi-static torque balance.

This term captures stability with respect to lateral loading and complements the angular interpretation of θ. Because appropriate stability requirements vary substantially across object categories, we normalize both quantities relative to the corresponding real product rather than imposing one universal target. We define

$$\theta^\star = \mathrm{clip}_{[5^\circ,30^\circ]}\left(\theta_{\mathrm{ref}}\right)$$

and

$$f^\star = \mathrm{clip}_{[0.05,0.35]}\left(f_{\mathrm{ref}}\right),$$

where θ_{ref} and f_{ref} are computed from the reference structure. The final reliability score is

$$S_{1.3} = \frac{1}{2}\left(\mathrm{clip}\left(\frac{\theta}{\theta^\star}\right) + \mathrm{clip}\left(\frac{f}{f^\star}\right)\right).$$

Reference-relative normalization is important because different artifact categories naturally exhibit different stability profiles. For example, a tall articulated structure and a wide low-profile object should not be expected to share the same tipping margin. Using the corresponding real product as a reference therefore provides a category-sensitive target. At the same time, clipping $\theta^\star$ and $f^\star$ prevents unusually stable or unusually unstable reference objects from defining excessively demanding or permissive evaluation criteria. The two terms also capture complementary failure modes. A structure with a narrow support footprint and a high center of mass may tip under only a small angular perturbation, while a tall structure may also be vulnerable to relatively small lateral forces applied near its top. Averaging the two normalized proxies reduces dependence on any single geometric characteristic and provides a more balanced measure of quasi-static stability. This metric is intentionally limited to tipping reliability. It does not attempt to model sliding, elastic deformation, dynamic impact, material fracture, joint strength, or fatigue, which require more detailed material models and dynamic simulation. These aspects are therefore outside the scope of the current geometric stability evaluation.

Evaluation Basis for Functional Affordance. Functional Affordance evaluates whether a generated structure contains the geometry, constituent components, and articulation needed to support the intended object category. The evaluation is fully deterministic and does not rely on perceptual judging. Instead, its evaluation targets are derived from structured functional knowledge extracted from Wikipedia together with class-level statistics obtained from real object instances. For each task, we construct a component sheet from required-part (P), functional-subsystem (F), and kinematic (K) claims. Canonical component phrases are normalized through tokenization, removal of positional modifiers, singularization, spelling normalization, and merging of equivalent alternatives. Roles in the generated structure are then grounded to these canonical components using deterministic phrase matching, predefined lexical mappings, and component-specific fallback rules. To avoid treating a single reference decomposition as the only valid realization of an object category, we estimate component relevance and expected multiplicity from a population consisting of the task reference and up to eleven additional real instances from the same object class. A component is retained as evaluation-relevant if it occurs in the reference or appears in at least 40% of the available population instances. For plural components, we define the minimum expected multiplicity as

$$m_c = \min\left(8,\ \max\left(1, \lfloor Q_{25}(\mathrm{count}_c)\rfloor\right)\right),$$

where Q_{25} denotes the lower quartile of the observed component counts. Using the lower quartile provides a conservative class-level expectation that accommodates legitimate variation across realizations, while the upper cap of eight prevents highly repetitive components from disproportionately dominating the evaluation. Singular components use $m_c = 1$.

Functional Affordance – A.1 Geometry

Functional objects must not only contain recognizable components, but should first, exhibit a global geometry consistent with the intended object category. Geometry therefore evaluates correspondence between the generated structure and the task reference at both the whole-object and, when semantic roles are available, component levels. Importantly, the metric is designed to

measure shape rather than absolute physical scale. We uniformly sample 20,000 surface points from both the generated design and the reference object. Each point cloud is independently centered using its horizontal bounding-box center and minimum height and then divided by its own bounding-box diagonal. Let p_τ denote the fraction of generated samples lying within distance τ of the reference surface and r_τ the corresponding fraction of reference samples lying within τ of the generated surface. We compute the bidirectional geometric F-score

$$F_\tau = \frac{2p_\tau r_\tau}{p_\tau + r_\tau}.$$

To account for ambiguity in the canonical front direction, we evaluate the design under four 90° yaw orientations and retain the best whole-object match:

$$F_{\text{shape}} = \max_{\psi \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} F_{0.025}\left(R_z(\psi)D,\, R\right).$$

When the design provides sufficiently informative part roles, we additionally evaluate component-level correspondence. A design is considered role-aware when at least 50% of its parts have non-generic semantic names. For each visible component c represented by at least 20 reference samples, we collect the generated samples belonging to parts whose roles ground to c and compute

$$F_c = F_{0.05}(D_c, R_c).$$

If the generated design contains no part grounded to component c, we set $F_c = 0$. Component comparison remains in the common normalized coordinate frame, such that the correct component geometry at an incorrect spatial location is appropriately penalized.
The final score is

$$A_{2.1} = \frac{0.6F_{\text{shape}} + 0.4\overline{F_c}}{0.6 + 0.4\,\mathbf{1}[\text{component term defined}]}.$$

Thus, role-less designs are evaluated solely on overall shape, while designs that explicitly expose semantic part structure are additionally evaluated on whether those components occupy geometrically appropriate regions. Created parts and library-retrieved parts are treated identically. The metric therefore does not reward a particular construction mechanism; it evaluates the geometry ultimately produced by that mechanism.

Functional Affordance – A.2 Parts

Global geometric similarity alone does not establish whether a generated object contains the distinct functional components expected of the target category. The Parts metric therefore evaluates both component coverage and multiplicity, while explicitly allowing valid generated decompositions to differ from the segmentation of any single reference instance. Generated parts are first assigned to canonical components. When an informative semantic role is available and can be grounded to a canonical component, this assignment is made directly from the declared role. Otherwise, we fall back to spatial correspondence: a generated part is assigned to the reference component that covers at least 50% of its surface samples within normalized distance 0.05, provided that at least five samples are available. This fallback allows systems without explicit semantic labels to remain evaluable while requiring substantial geometric agreement before a component identity is inferred. Because a single functional component may itself be represented by multiple generated parts, parts assigned to the same canonical component are grouped into a common instance when they are connected by fixed joints or when their axis-aligned bounding boxes overlap by more than 20% of the smaller box. This prevents decomposition granularity alone from affecting component counts: for example, a wheel represented by separate rim, tire, and hub geometries may still correspond to a single functional wheel instance. Reference segmentations do not necessarily expose all legitimate internal functional structure. To avoid systematically penalizing such alternative realizations, we additionally identify anonymous interior modules. The generated object is voxelized using

$$r_{\text{int}} = \text{clip}\left(\frac{\text{diag}(D)}{150}, 0.5\text{ mm}, 4\text{ mm}\right),$$

and exterior air is recovered through a 6-connected flood fill. An unassigned, non-brick part is treated as interior when at most 5% of its boundary cells are adjacent to exterior air. Parts satisfying this criterion are grouped into anonymous internal modules. This mechanism does not assume that such modules correspond to a particular semantic component; rather, it provides conservative evidence that the generated design contains nontrivial internal structure that may implement a required but unobserved function. For each canonical component c, an identified instance receives score 1 when it participates in the connected assembly and score $1/2$ when all of its constituent parts are floating. This distinction ensures that mere geometric presence is not treated as equivalent to functional integration into the artifact. Given the task-specific minimum expected count m_c, the component score is

$$s_c = \frac{\sum_{j=1}^{m_c} s_c^{(j)}}{m_c},$$

where $s_c^{(j)}$ are the m_c highest instance scores. Thus, a design receives full credit only when the required number of component instances is present and integrated, while additional redundant instances do not inflate the score. For a non-visible component for which no explicit instance can be identified, one anonymous interior module may be consumed to provide partial credit of $1/2$. This deliberately conservative treatment acknowledges plausible hidden functional structure without equating an unlabeled internal module with an explicitly identified component.
The final Parts score is

$$A_{2.2} = \frac{2}{3}\overline{s_c}^{\,\text{visible}} + \frac{1}{3}\overline{s_c}^{\,\text{internal}},$$

with the weights renormalized when either group is empty. The larger weight on visible components reflects the stronger evidence available for externally identifiable structure, while the internal term preserves sensitivity to hidden functional organization. Overall, the metric is designed to evaluate whether the expected functional parts are present in sufficient number and meaningfully integrated, without requiring the generated artifact to reproduce a particular reference decomposition.

Functional Affordance – A.3 Kinematics

Many buildable objects are defined not only by the presence and arrangement of their parts, but also by the relative motions that enable their intended function. Kinematics therefore evaluates whether generated moving joints provide the expected motion type, orientation, placement, range, moving-part extent, and collision-free articulation required by the target object. The metric is deliberately stricter than simply checking whether a non-fixed joint has been declared: a useful articulation must be both semantically appropriate and geometrically realizable. Kinematic targets are constructed from two complementary sources. First, each non-fixed reference articulation whose moving child can be grounded to a canonical component defines a reference-based motion target. Revolute, continuous, cylindrical, and screw joints are grouped as rotational motion; prismatic joints define translational motion; and ball joints define spherical motion. Second, each curated K claim whose moving phrase grounds to a canonical component defines a claim-based target. When a claim describes the same articulation as a reference joint, the two are merged into a single target rather than counted independently. Claims associated with non-visible components are treated as **declared-only**: they may justify a generated joint when computing precision, but they do not introduce an additional recall requirement. This distinction prevents incomplete observability from creating unsupported mandatory motions while still allowing semantically plausible articulations to be recognized. Where sufficient real-instance evidence is available, expected kinematic properties are derived from the population rather than imposed as universal assumptions. These properties include the permissible joint type, whether the motion axis should align with a symmetry or principal axis, whether its direction should be vertical or horizontal, whether the joint should lie near the component center or edge, its expected angular span or translation ratio, and the maximum plausible fraction of the artifact that should move with the joint. When population evidence is insufficient, the corresponding generic expectation is used. This combines category-specific evidence with

deterministic fallback rules, allowing the metric to accommodate heterogeneous object classes without requiring a single kinematic template for all structures. For a generated joint j, we define joint quality as

$$q_j = q_j^{\text{type}} q_j^{\text{axis}} q_j^{\text{offset}} q_j^{\text{range}} q_j^{\text{moveset}} q_j^{\text{free}}.$$

The multiplicative form is intentional: these terms represent complementary conditions that are jointly required for a useful articulation, such that a severe failure in any one dimension should substantially reduce the validity of the overall motion. The type term is 1 for a compatible joint, 0.75 when a cylindrical joint is used for an otherwise compatible rotational or translational target, and 0 for an incompatible type. Axis consistency is evaluated using

$$g(e) = \text{clip}\left(\frac{30^\circ - e}{20^\circ}, 0, 1\right),$$

where e denotes the relevant angular deviation from the expected symmetry, principal, vertical, or horizontal axis. The continuous falloff avoids introducing a brittle binary decision near a single angular threshold while still assigning no credit to substantially misaligned motion axes. Joint placement is evaluated relative to the geometry of the corresponding component. Let o denote the distance from the component-instance center to the joint axis, normalized by the component extent in the relevant direction. A center-oriented joint receives

$$q_j^{\text{offset}} = \text{clip}\left(\frac{0.7 - o}{0.4}, 0, 1\right),$$

whereas an edge-oriented joint receives

$$q_j^{\text{offset}} = \text{clip}\left(\frac{o - 0.2}{0.3}, 0, 1\right).$$

Offsets larger than 1.5 receive zero credit. These two cases capture common functional distinctions such as rotation about a wheel center versus articulation about the edge of a door-like component. The range term evaluates whether the declared motion is sufficiently large to support the intended articulation. For rotational targets,

$$q_j^{\text{range}} = \min\left(1, \frac{\text{span}_j}{\text{span}^\star}\right),$$

while for translational targets,

$$q_j^{\text{range}} = \min\left(1, \frac{|\Delta \text{limit}_j|}{\rho^\star L_{\text{move}}}\right),$$

where $\rho^\star$ is the expected travel ratio and L_{move} is the moving set extent along the joint axis. Ball joints receive full range credit. These ratios measure whether a declared joint merely permits motion in principle or provides a motion extent consistent with the intended function.

The moving set contains the joint child together with geometry reachable through subsequent joints and, for otherwise joint-less parts, physical contact, without traversing through the parent. A motion receives no moving set credit if this set contains the target's relative component or degenerates into motion of the entire object. Otherwise, the score decreases linearly when the moving fraction exceeds the target-specific expected maximum. This term is important because a formally valid joint is not functionally meaningful if it causes both sides of the intended articulation—or nearly the entire artifact to move together. Finally, q_j^{free} evaluates whether the declared articulation is geometrically realizable rather than merely specified in metadata. We voxel-sweep the moving set through multiple poses over the declared motion range and measure the fraction of poses that do not introduce substantial deep overlap with static geometry. Full rotations use 11 poses at 30° increments; other motions use eight poses over their declared interval. A moving set that is already deeply embedded in static geometry at rest receives zero free-motion credit. This discrete sweep is intended as a deterministic geometric feasibility test rather than a full dynamic simulation. We aggregate joint-level measurements through both recall and precision. Each required target takes the best joint quality obtained by each of its component instances, and the target receives the average of its best m_c instances. Averaging

over all targets gives recall R, which measures whether the expected articulations are successfully realized. For precision, every generated moving joint receives the best quality it achieves on any required target or declared-only claim. A joint unmatched to any such target may still receive half of its quality under the generic expectation for its own joint type, allowing mechanically plausible but unclaimed articulations to receive limited credit without being treated as equivalent to expected motions. Letting P denote the mean score over generated moving joints, the final metric is

$$A_{2.3} = \frac{2RP}{R+P}.$$

If no moving joint is generated, $P = 0$. The harmonic aggregation prevents systems from obtaining a high score either by omitting required articulations or by introducing large numbers of unsupported moving joints. The metric is not applicable to tasks for which no kinematic target exists; in the current benchmark, this only includes: `bed-frame`, `dining-chair`, `dining-table`, and `step-ladder`.

Design Quality: Perceptual Evaluation and Human Calibration.

Design Quality captures aspects of generated structures for which no single deterministic geometric criterion provides an adequate evaluation. We therefore use a **Calibration-then-Evaluation** protocol: human judgments on a fixed, pre-declared calibration subset are used only to calibrate the scale and relative contribution of automated evaluation signals; the resulting calibration parameters are then frozen and applied uniformly to the remaining benchmark outputs.

Our automatic perceptual signal is obtained from two independent VLM judges, Qwen2.5-VL-32B-Instruct and Gemma-3-27B-IT. Each judge is provided with explicitly defined, metric-specific judging criteria and assigns a raw score $s \in 1, \ldots, 10$ to each criterion. We normalize each raw judgment as $(s-1)/9$. Let $J \in [0,1]$ denote the mean of these normalized scores across the defined criteria and the two VLM judges. Where an objective geometric or programmatic measurement is available, we denote the corresponding computed signal by C. Thus, for a given Design Quality metric, the available automated evaluation signals are drawn from $h \in J, C$.

Because the raw perceptual and computed signals need not naturally share the same scale as human judgments, each available signal h is independently calibrated. Let h' denote its calibrated value, and let ℓ_h and u_h denote the raw values of h that map to calibrated scores 0 and 1, respectively. We define

$$h' = \mathrm{clip}\left(\frac{h-\ell_h}{u_h-\ell_h}, 0, 1\right).$$

Human calibration targets are collected on a 1–5 rating scale. Let $r \in 1, \ldots, 5$ denote a raw human rating, and let

$$H = \frac{r-1}{4}$$

denote its normalized value in $[0,1]$. To determine the calibration range for each automated signal h, we fit an affine mapping from h to the human score distribution. Let b denote the fitted scale factor and a the fitted offset. These parameters are obtained by matching the mean and standard deviation of the transformed automated scores to those of the normalized human ratings:

$$b = \frac{\mathrm{sd}(H)}{\mathrm{sd}(h)}, \qquad a = \overline{H} - b\overline{h}.$$

The corresponding raw-score endpoints that map to calibrated values 0 and 1 are then

$$\ell_h = -\frac{a}{b}, \qquad u_h = \frac{1-a}{b}.$$

Accordingly, ℓ_h and u_h define the metric-specific calibration interval for signal h.

Calibration is fit on 48 pre-declared items spanning eight objects. When both perceptual and computed signals are available, we combine their calibrated values using nonnegative weights α and β, where α controls the contribution of the perceptual signal J' and β controls the contribution of the computed signal C'. The ratio $\alpha : \beta$ is selected on the calibration set using leave-one-object-out mean absolute error (MAE), such that all items associated with the held-out object are excluded when evaluating each candidate blending ratio. After this fitting stage, the calibration ranges and blending weights are fixed and reused unchanged for benchmark evaluation.

The resulting metric-specific calibration parameters are

Metric	$\ell_J \to u_J$	$\ell_C \to u_C$	$\alpha : \beta$
$D.1$	$0.3611 \to 0.7644$	$0.1276 \to 1.0247$	$1:1$
$D.2$	$0.3741 \to 0.7251$	–	$1:0$
$D.3$	$0.4500 \to 0.7766$	$0.1536 \to 0.6057$	$3:1$

where J' and C' denote the calibrated perceptual and computed signals obtained from the transformation above. When both signals are available, the final Design Quality metric is

$$D = \frac{\alpha J' + \beta C'}{\alpha + \beta}.$$

When only one signal is defined, its calibrated value is used directly.

Human Annotation Details for Design Evaluation. We recruited four university lab volunteers to annotate 90 objects generated by six systems. Annotators provided 1–5 ratings for decomposition quality, aesthetics, and reference alignment. We used human ratings from a subset of eight objects to calibrate our Design evaluation metrics, and evaluated the resulting metrics on a disjoint held-out set, as reported in Table 5. The calibrated metrics show strong agreement with human judgments at the system level, with Spearman correlations of 0.77, 0.83, and 0.94 for decomposition, aesthetics, and alignment, respectively. Item-level correlations are more moderate but remain positive across all three dimensions. Overall, these results suggest that our Design metrics capture human-perceived differences between systems reasonably well, while still leaving room for disagreement at the individual-object level.

Design Quality – D.1 Decomposition

A buildable structure should be decomposed into components that correspond to meaningful and independently realizable parts, rather than represented as either a single monolithic mesh or an arbitrary collection of fragments. At the same time, many real objects admit multiple valid decompositions, and the segmentation of a single reference instance should not be treated as the unique ground truth. Decomposition therefore combines a perceptual assessment of whether the proposed part organization is meaningful with conservative programmatic signals that capture specific decomposition failure modes. The perceptual and computed signals are calibrated independently using the common Calibration-then-Evaluation protocol described above and are then combined using the pre-determined D.1 blending ratio.

Perceptual decomposition quality. To expose the generated part organization to the perceptual judges, each part is rendered using a distinct color so that part boundaries remain visually identifiable. For designs containing at most 40 parts, the decomposition sheet contains two assembled views and two exploded views, with each part displaced outward from the object center by 0.25 times the object diagonal. For designs containing more than 40 parts, non-exploded front and top views are used instead to avoid excessive visual clutter. The judges additionally receive the task condition image for semantic context and are explicitly informed of the generated part count.
Each judge evaluates three complementary criteria: **boundary quality**, **granularity**, and **separability**. Boundary quality measures whether part divisions occur at structurally or semantically meaningful locations. Granularity measures whether the decomposition uses an appropriate level of detail rather than being excessively fragmented or overly coarse. Separa-

bility measures whether the resulting components correspond to plausible independently realizable parts. Let $x_{q,c} \in 1, \ldots, 10$ denote the raw score assigned by judge q to criterion c. Let $\mathcal{C} = \text{boundary}, \text{granularity}, \text{separability}$ denote the three perceptual criteria, and let $x_{q,c} \in 1, \ldots, 10$ be the score assigned by judge q to criterion c. The perceptual decomposition signal is

$$J_{3.1} = \frac{1}{6} \sum_{q \in \{\text{Qwen}, \text{Gemma}\}} \sum_{c \in \mathcal{C}} \frac{x_{q,c} - 1}{9}.$$

Programmatic decomposition quality. We complement the perceptual assessment with a computed signal $C_{3.1}$ consisting of the programmatic terms that are defined for the generated design. These terms are not intended to prescribe a unique decomposition; instead, they provide conservative evidence for three concrete failure modes: substantial disagreement in decomposition granularity, unnecessary fragmentation of otherwise continuous components, and geometrically unsupported joint placement.

The first term measures reference-relative decomposition granularity. Let n denote the number of generated parts and n_{ref} the number of parts in the segmented reference. We define

$$C_{\text{gran}} = \frac{\min(n, n_{\text{ref}})}{\max(n, n_{\text{ref}})} = 2^{-|\log_2(n/n_{\text{ref}})|}.$$

This formulation symmetrically penalizes proportional over- and under-decomposition: for example, generating twice as many parts as the reference receives the same score as generating half as many. Importantly, the reference part count serves only as supporting evidence of decomposition granularity rather than as a required target, since alternative physically meaningful realizations may legitimately contain different numbers of components. The second term detects unnecessary fragmentation of a component that otherwise appears continuous. This signal is defined when material declarations are available. Let V_{split} denote the number of interfacing pairs $(i, k) \in \mathcal{I}$ that simultaneously satisfy three conditions: the two parts ground to the same canonical semantic role, use the same material, and have no declared joint between them. Such a configuration provides evidence that geometry corresponding to a common functional component may have been unnecessarily divided. We define

$$C_{\text{consol}} = \text{clip}\left(1 - \frac{V_{\text{split}}}{\max(n-1, 1)}, 0, 1\right).$$

The normalization by $\max(n-1, 1)$ scales the penalty relative to the size of the generated decomposition, while clipping ensures that the term remains within the common $[0, 1]$ score range even when the number of flagged interfacing pairs exceeds $n-1$. This can occur because a part may interface with multiple neighboring parts. The criterion is intentionally conservative: geometric contact alone is insufficient to identify unnecessary fragmentation. Parts must additionally share the same canonical semantic role and material, and the absence of a declared articulation prevents intentionally movable interfaces from being penalized. The third term evaluates whether declared joints are supported by the corresponding geometric interfaces. Among interfacing part pairs carrying a declared joint, let N_{decl} denote the number of such declarations and N_{rej} the number that fail the shared fastening criterion defined in the common evaluation conventions. When at least one such declaration is available, we compute

$$C_{\text{joint}} = 1 - \frac{N_{\text{rej}}}{N_{\text{decl}}}.$$

This term prevents an implausible decomposition from receiving support merely through arbitrary joint declarations: a declared connection contributes positively only when its placement and relationship to the incident parts are geometrically supported. When no applicable joint declaration exists, this term is treated as unavailable rather than as evidence for or against the decomposition. The raw computed decomposition signal is the mean of the programmatic terms available for the design:

$$C_{3.1} = \text{mean}\left(C_{\text{gran}}, C_{\text{consol}}, C_{\text{joint}}\right)_{\text{available}}.$$

Calibration and aggregation. The perceptual signal $J_{3.1}$ and computed signal $C_{3.1}$ are independently calibrated to the human rating scale using the common calibration procedure described above, producing $J' * 3.1$ and $C' * 3.1$. The calibration parameters and blending ratio are determined during the pre-declared calibration stage and remain fixed during benchmark evaluation. As established by this procedure, D.1 assigns equal weight to the perceptual and computed signals:

$$D_{3.1} = \frac{J'_{3.1} + C'_{3.1}}{2}.$$

When only one signal is available, its calibrated value is used directly. This hybrid formulation deliberately avoids treating the segmentation of a single reference object as the unique correct decomposition. Instead, the perceptual component evaluates whether the proposed organization forms meaningful and independently realizable parts, while the programmatic component provides complementary evidence for specific structural failure modes, including severe granularity mismatch, unnecessary fragmentation, and unsupported joint placement.

Design Quality – D.2 Aesthetics

Aesthetics evaluates the intrinsic visual quality of the generated structure independently of its exact correspondence to a particular reference instance. This distinction is important because buildable objects may admit substantial variation in valid design: a generated artifact may differ from the reference while still exhibiting well-resolved proportions, coherent form, and high geometric refinement. Conversely, approximate reference similarity does not by itself imply a visually well-resolved design. To isolate geometric form from superficial appearance, all generated components are rendered using the same neutral gray material. Colors, textures, semantic labels, and material assignments therefore do not contribute to this metric, and individual part boundaries are intentionally deemphasized. The judges receive only the generated clay renderings and are not shown the reference object or task condition image. Aesthetics is therefore evaluated as a reference-free property of the generated design. Each judge evaluates three criteria: **proportion**, **coherence**, and **refinement**. Proportion measures whether the relative dimensions and scales of the visible forms produce a plausible and balanced whole. Coherence measures whether the geometry reads as a unified design rather than as an arbitrary collection of independently generated shapes. Refinement measures whether the major forms, transitions, and characteristic geometric details are sufficiently resolved rather than remaining coarse or unfinished. Let

$$\mathcal{Q} = \{\text{Qwen}, \text{Gemma}\}, \qquad \mathcal{C}_{\text{aest}} = \{\text{proportion}, \text{coherence}, \text{refinement}\},$$

and let $x_{q,c} \in 1, \ldots, 10$ denote the score assigned by judge $q \in \mathcal{Q}$ to criterion $c \in \mathcal{C}_{\text{aest}}$. The raw perceptual signal is

$$J_{3.2} = \frac{1}{6} \sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}_{\text{aest}}} \frac{x_{q,c} - 1}{9}.$$

Unlike D.1 and D.3, Aesthetics deliberately contains no separate programmatic component. Simple geometric proxies such as reference overlap, symmetry, or part count would impose assumptions about what a visually successful design should look like and would partially conflate intrinsic design quality with reference fidelity. D.2 therefore follows the same Calibration-then-Evaluation protocol using only its perceptual signal: $J_{3.2}$ is calibrated to the human rating scale using the pre-declared calibration set, and the resulting calibration parameters are subsequently frozen for benchmark evaluation. The final score is

$$D_{3.2} = J'_{3.2}.$$

This metric is consequently reference-free by design. It rewards generated artifacts whose geometry is proportionally plausible, visually integrated, and sufficiently refined, without requiring reproduction of the particular design choices present in the benchmark reference.

Design Quality – D.3 Structural Alignment

Structural Alignment evaluates whether the generated artifact preserves the characteristic organization and design-defining structure expressed by the task condition. Unlike D.2 Aesthetics, which evaluates the generated object independently of any reference, D.3 is explicitly reference-conditioned. At the same time, it does not require exact geometric reconstruction: multiple valid realizations may preserve the same characteristic design while differing in detailed geometry, construction strategy, or decomposition. We therefore combine perceptual reference-based assessment with conservative programmatic signals of structural correspondence. D.3 is also intentionally distinct from A.1 Geometry. A.1 primarily measures general geometric correspondence at the whole-object and component levels, asking whether the generated geometry broadly matches the intended object and occupies appropriate spatial regions. In contrast, D.3 focuses on finer-grained, design-specific correspondence that is difficult to capture with geometry-only rules, including whether characteristic components are represented, whether their relative arrangement and scale preserve the target design, and whether distinctive structural details remain recognizable. These properties often depend on semantic and perceptual interpretation rather than distance alone. Accordingly, D.3 uses VLM judgment as its primary signal, while programmatic measurements provide complementary geometric evidence rather than replacing perceptual assessment.

Perceptual structural alignment. For the perceptual component, the judges receive both the task condition image and the neutral gray clay rendering of the generated structure. Each judge evaluates four criteria: **inventory**, **placement**, **relative scale**, and **detail fidelity**. Inventory measures whether the major characteristic components of the target are represented. Placement measures whether these components occupy appropriate relative locations. Relative scale measures whether their sizes and proportions preserve the target organization. Detail fidelity measures whether distinctive geometric features and structural details remain recognizable in the generated design. Let

$$\mathcal{Q} = \{\mathrm{Qwen}, \mathrm{Gemma}\}, \qquad \mathcal{C}_{\mathrm{align}} = \{\mathrm{inventory}, \mathrm{placement}, \mathrm{scale}, \mathrm{detail}\},$$

and let $x_{q,c} \in 1, \ldots, 10$ denote the score assigned by judge $q \in \mathcal{Q}$ to criterion $c \in \mathcal{C}_{\mathrm{align}}$. The raw perceptual signal is

$$J_{3.3} = \frac{1}{8} \sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}_{\mathrm{align}}} \frac{x_{q,c} - 1}{9}.$$

Programmatic structural alignment. We complement the perceptual assessment with up to two programmatic signals. These signals are intended as supporting evidence for structural correspondence and do not impose exact reconstruction of the reference.

The first signal measures part-level volumetric correspondence. The generated design and reference object are independently normalized for scale using their respective bounding-box diagonals and placed in the common normalized comparison frame. Each generated and reference part is then voxelized from surface samples followed by hole filling, using a voxel pitch equal to 1% of the corresponding object's diagonal. For each reference part r, we identify the generated part p with maximum volumetric intersection-over-union. The resulting score is

$$C_{\mathrm{part}} = \frac{\sum_r |r| \max_p \mathrm{IoU}(r, p)}{\sum_r |r|},$$

where $|r|$ denotes the occupied volume of reference part r.

The matching is intentionally many-to-one rather than one-to-one. This allows alternative valid decompositions in which multiple reference segments are represented by a single generated component. Weighting by reference-part volume gives greater influence to major structural components while preventing very small decorative pieces from dominating the score. No additional rotation search is performed for this term, so correspondence is evaluated in the shared task-aligned coordinate frame.

The second programmatic signal evaluates consistency in global mirror symmetry when the corresponding reference symmetry measurement is available. For each object, occupied geometry is represented on a 1 cm voxel grid. We evaluate 36 vertical mirror planes passing through

	Decomposition		Aesthetics		Alignment	
System	Human	Ours	Human	Ours	Human	Ours
Claude Sonnet 5	92.9 (1)	84.9 (1)	76.8 (2)	79.6 (1)	80.4 (1)	74.4 (2)
GPT-6 Astra	83.9 (2)	80.0 (2)	85.7 (1)	72.0 (3)	78.6 (2)	78.9 (1)
PartPacker	69.6 (3)	59.4 (5)	60.7 (3)	78.1 (2)	60.7 (3)	71.4 (3)
Qwen3.5-27B	50.0 (5)	76.6 (3)	44.6 (4)	66.0 (4)	46.4 (4)	65.8 (4)
Gemma-4-31B	60.7 (4)	60.2 (4)	32.1 (5)	54.9 (5)	37.5 (5)	62.0 (5)
BrickGPT	3.6 (6)	13.0 (6)	23.2 (6)	12.2 (6)	8.9 (6)	35.8 (6)
System Spearman $\rho \uparrow$	0.77		0.83		0.94	
System MAE $\downarrow$	9.8		14.8		14.6	
Item Spearman $\rho \uparrow$	0.52		0.60		0.64	
Human–human ρ (ref.)	0.69		0.52		0.73	

Table 5: Alignment between calibrated Design (D) scores and human ratings on 7 held-out objects (6 systems, 42 designs per dimension; 4 annotators). Cells report system mean scores with ranks in parentheses. We report system-level Spearman ρ and MAE, item-level Spearman ρ, and human–human ρ as a reference for annotator agreement.

the object's centroid, separated by 5° increments, and define the object's symmetry value as the maximum IoU between its occupied cells and their reflected counterparts. Let $\sigma \in [0, 1]$ and $\sigma_{\text{ref}} \in [0, 1]$ denote the resulting symmetry values for the generated and reference structures, respectively. We define

$$C_{\text{sym}} = 1 - \frac{|\sigma - \sigma_{\text{ref}}|}{\max(\sigma_{\text{ref}}, 1 - \sigma_{\text{ref}})}.$$

Importantly, C_{sym} does not reward symmetry itself. It measures agreement with the symmetry characteristics of the target. A strongly asymmetric reference therefore does not require the generated artifact to become symmetric, whereas a highly symmetric reference provides evidence that comparable global organization is characteristic of the target design.
The raw computed signal is the mean of whichever programmatic terms are available:

$$C_{3.3} = \text{mean}\,(C_{\text{part}}, C_{\text{sym}})_{\text{available}}\,.$$

For tasks where reference geometry is available but no valid symmetry measurement is provided, C_{part} remains defined and constitutes the computed component on its own.

Calibration and aggregation. The perceptual signal $J_{3.3}$ and computed signal $C_{3.3}$ are independently calibrated to the human rating scale using the common Calibration-then-Evaluation procedure, producing $J' * 3.3$ and $C' * 3.3$. The calibration parameters and blending ratio are determined during the pre-declared calibration stage and remain fixed during benchmark evaluation. As established by this procedure, D.3 assigns a 3:1 perceptual-to-computed weighting:

$$D_{3.3} = \frac{3J'_{3.3} + C'_{3.3}}{4}.$$

This weighting reflects the role of D.3 as a fine-grained structural and design-alignment measure: perceptual judgment captures semantic organization and distinctive design features that are difficult to express through deterministic geometric rules alone, while the computed component provides complementary evidence of part-level correspondence and, where available, symmetry consistency. In this way, D.3 complements rather than duplicates A.1: A.1 measures broad geometric correspondence, whereas D.3 evaluates whether the characteristic structural identity of the specific target design is preserved.

Physical Realization: Evaluation Basis. The preceding metric levels evaluate whether a generated structure is geometrically sound, contains the functional components expected of the target object, and exhibits coherent design quality. Physical Realization asks a complementary question: whether the generated specification provides a plausible path from a digital assembly to an artifact that could be constructed and used. We evaluate three complementary aspects. **Assembly Sequence** examines

whether the parts can be inserted in the declared order while maintaining physically supportable intermediate states. **Material** evaluates whether declared material choices are compatible with the functional role of each component. **Operability** evaluates whether the completed structure supports the required functional subsystems, articulated motions, and human-use conditions.

These metrics therefore evaluate information contained in the generated assembly specification rather than only the final rendered geometry. Missing assembly instructions, missing materials, absent semantic roles, or unsupported articulation may reduce realization performance even when the final shape appears visually plausible.

Physical Realization – R.1 Assembly Sequence

A physically plausible final geometry is not necessarily assemblable. A component may be trapped by previously placed geometry, an intermediate subassembly may remain unsupported or become unstable before later components are added, or the declared construction sequence may omit parts that appear in the final design. Assembly Sequence therefore evaluates both whether the declared ordering admits physically feasible insertion operations and whether the resulting intermediate configurations remain supportable throughout construction. Let the declared sequence contain m placement operations,

$$\{(p_k, d_k)\}_{k=1}^{m},$$

where p_k denotes the component placed at step k and d_k its declared insertion direction. Let n denote the total number of parts in the final generated design, including any parts omitted from the declared sequence. A design for which no usable assembly sequence is provided receives a score of zero. After each placement, the currently assembled structure is partitioned into connected subassemblies according to the shared **physical-connectivity** relation defined above, which recognizes geometrically interlocked parts, valid fastening relations, and valid brick–stud connections. Mere geometric contact alone is therefore insufficient to merge two parts into the same subassembly. Each step is then evaluated along two dimensions: **insertability**, which tests whether the newly introduced component can reach its final pose without passing through previously assembled geometry, and **intermediate support**, which tests whether the resulting configuration remains physically grounded and, where applicable, stable against tipping.

Insertability. For component p_k, the declared direction d_k is evaluated first as the preferred insertion hypothesis. We additionally test the fixed canonical directions

$$-z, \quad \pm x, \quad \pm y, \quad +z$$

as deterministic fallbacks. Thus, the metric evaluates whether the declared assembly order admits a physically plausible insertion path, rather than requiring exact agreement with a single insertion-direction annotation. For each candidate direction, the component is swept from outside the current object extent toward its final pose in discrete odd-cell increments over a search distance extending to the object bounding-box diagonal plus 5 cm. Let O_t denote its voxel overlap with previously placed geometry at an intermediate sweep position and O_{final} the corresponding overlap at the final pose. The insertion path is accepted only when

$$O_t \leq O_{\text{final}} + 2$$

throughout the sweep. This relative criterion tolerates small discretization or interface overlap already present in the intended final configuration while rejecting paths that require the component to pass substantially through existing geometry. Because the supporting floor is not represented as collision geometry, upward insertion directions with vertical component $d_z > 0.3$ are excluded when the component's final pose lies above the ground-contact band. This prevents artificial assembly paths in which a component would effectively enter the structure from below the supporting surface. The first component in the sequence is considered insertable by definition.

Intermediate support and stability. After a successful insertion test, the connected subassembly containing the newly placed component must reach the common ground-contact band,

$$z_{\min} \leq z_0 + \beta,$$

where z_0 and β follow the same ground-contact convention used in S.3. For free-standing tasks, each intermediate subassembly must additionally satisfy positive quasi-static tipping stability. Let X_k denote the set of connected subassemblies present after step k, let θ_B denote the critical tilt angle of subassembly $B \in X_k$, and let $\theta^\star$ denote the task-specific stability target defined in S.3. We define the intermediate-state margin as

$$s_k = \min_{B \in X_k} \min \left(1, \frac{\theta_B}{\theta^\star}\right).$$

The minimum over subassemblies reflects the fact that a single unsupported or unstable intermediate component is sufficient to compromise the current assembly state. If step k is not executable, or if any resulting subassembly is floating or self-tipping, we set

$$s_k = 0.$$

For tasks whose references are not free-standing, the stability requirement reduces to binary grounding, and therefore $s_k \in 0, 1$. A component associated with a failed step remains in the simulated assembly, and later declared steps are still evaluated. This is intentional: the executable-prefix term described below records the first break in sequential feasibility, whereas the intermediate-state term retains information about whether subsequent portions of the declared construction trajectory remain locally plausible.

Aggregation. Let $k^\star$ denote the number of consecutive assembly steps successfully completed before the first failure. The executable-prefix score is

$$E = \frac{k^\star}{n}.$$

The intermediate-state score is

$$S = \frac{1}{n} \sum_{k=1}^{m} s_k.$$

Both terms are normalized by the total number of parts n in the generated design rather than by the number of declared steps m. Consequently, omitting difficult components from the assembly sequence cannot improve the score.
The final Assembly Sequence metric is

$$R_{4.1} = \frac{1}{2}E + \frac{1}{2}S.$$

The two terms capture complementary aspects of constructability: E measures how much of the intended artifact can be assembled sequentially before the first infeasible operation, while S measures the quality and supportability of the intermediate configurations encountered across the declared construction process.

Physical Realization – R.2 Material

Physical realization additionally requires material assignments that are compatible with the structural and functional role of each component. Because real products frequently admit multiple valid material choices, Material does not evaluate exact agreement with the material of a single reference model. Instead, for each component we construct a set of acceptable material classes from multiple independent evidence sources and evaluate whether the generated assignment is compatible with that set. This formulation is intentionally permissive with respect to legitimate material variation while remaining restrictive toward assignments that are unsupported by either documented usage, observed population statistics, reference correspondence, or functional plausibility. Our material library contains 23 materials organized into nine classes. For compatibility reasoning, related classes are additionally grouped into broader families, including metal, polymer, soft material, wood, glass, and ceramic. For a generated part i with canonical semantic role r_i, the accepted material set is

$$\mathcal{A}_i = W(r_i) \cup P(r_i) \cup R(i),$$

where W, P, and R denote textual evidence, population statistics, and reference-overlap evidence, respectively. We use the union because the three sources are complementary and individually incomplete: a material may be well supported by one source while being absent from another simply because that source does not exhaust all valid realizations. Requiring agreement across sources would therefore be unnecessarily restrictive. If this union is empty, we instead use the functional material judgment

$$\mathcal{A}_i = J(r_i).$$

The first source, $W(r_i)$, is derived from verified material claims in the cited Wikipedia sources. We retain only claims that explicitly name the corresponding component. Per-part material extraction is performed from the cited text, and an extracted answer is accepted only when accompanied by a verbatim supporting quotation containing each relevant material expression. This requirement makes the textual evidence auditable and prevents unsupported material inference. Material claims associated with clearly historical usage, such as descriptions explicitly marked by historical years or terms such as 'originally" or 'traditionally," are excluded so that the metric better reflects plausible contemporary construction rather than historically possible but currently atypical realizations. The second source, $P(r_i)$, is a population prior estimated from Artiverse instances in the corresponding object category. A material class is included when it accounts for at least 10% of the observed class mentions for the role, provided that the role has at least five observations. The 10% threshold is intended to retain recurring alternatives while excluding isolated or exceptionally rare assignments, and the minimum observation count avoids drawing population-level conclusions from insufficient evidence. This permits multiple commonly observed materials to be treated as valid rather than forcing the design toward a single modal material. The third source, $R(i)$, transfers material evidence from geometrically corresponding reference components. The generated and reference objects are independently normalized by their diagonals and voxelized at a pitch equal to 2% of the corresponding diagonal. For every reference component that covers at least 25% of the generated part's occupied cells, we add the reference component's interior and exterior material classes together with the population prior associated with its semantic role. The overlap threshold requires substantial geometric correspondence before material evidence is transferred, reducing the chance that incidental contact or weak spatial proximity incorrectly determines the acceptable material set. When none of these three sources provides an applicable class, we use $J(r_i)$, a fixed task-and-part functional material table. This fallback represents material classes that are functionally plausible for the corresponding component role. It is used only when evidence from text, population statistics, and reference correspondence is unavailable, thereby providing coverage without allowing the fallback to override stronger task-specific evidence. Let c_i denote the library material class assigned to generated part i. Its per-part compatibility score is

$$a_i = \begin{cases} 1, & c_i \in \mathcal{A}_i, \\ 0.5, & c_i \text{ belongs to the same material family as a class in } \mathcal{A}_i, \\ 0, & \text{otherwise.} \end{cases}$$

A part with no declared library material receives zero whenever an evaluation basis exists. Exact class agreement therefore receives full credit, while a materially related choice receives partial credit. The family-level score of 0.5 should be interpreted as coarse compatibility rather than equivalence: it acknowledges that materials within the same broad family may share relevant physical characteristics while still distinguishing them from explicitly supported classes. For example, two metal subclasses may be treated as more similar than a metal and a textile even when the exact preferred class differs. The final Material score is

$$R_{4.2} = \frac{1}{|\{i : \mathcal{A}_i \neq \varnothing\}|} \sum_{i:\mathcal{A}_i \neq \varnothing} a_i.$$

Parts for which no evaluation basis can be established are excluded rather than automatically treated as incorrect. This conservative treatment avoids penalizing the generator when the

benchmark itself lacks sufficient evidence to determine material suitability. If no part has an available material basis, the metric is considered not applicable. Conversely, a generated design in which no part declares a material from the supported library receives a score of zero. Overall, this evidence-based formulation is designed to balance two competing requirements: it accommodates genuine material diversity across real products while still rejecting choices that are incompatible with the structural or functional role of the corresponding component. The numerical thresholds used for evidence inclusion and partial compatibility are fixed benchmark-wide heuristic choices, selected to provide stable and interpretable behavior across heterogeneous object categories rather than to reproduce any single reference construction exactly.

Physical Realization – R.3 Operability

A completed structure may be geometrically plausible and physically assemblable while still failing to support its intended use. Operability therefore evaluates whether the generated artifact can realize the task-level capabilities associated with the corresponding real object. Rather than treating functionality as a single visual or geometric property, each capability is represented as an explicit combination of required components, structural relationships, articulated behavior, and, where applicable, human-use constraints. R.3 consequently evaluates whether these individually necessary conditions are integrated into a structure that can plausibly operate as intended.

Capability specification. For each task, we construct a fixed capability sheet from the curated functional and kinematic claims together with a pre-declared human-use layer. The capability specification is task-level and system-independent: it is defined independently of the generated outputs and applied uniformly to all evaluated systems. Capabilities draw from four sources. Functional capabilities, denoted F, correspond to required functional subsystems of the object. Motion capabilities, denoted M, represent articulated behaviors required by the task. Human-use capabilities, denoted H, capture geometric requirements for interaction, including reachability, clearance, graspability, and usable support surfaces. Finally, P contains required components not already represented by a functional subsystem. A functional subsystem may consist of multiple required links. Each named component must be present, and explicitly specified component relationships must be supported by a contact or joint path of at most four hops. When the subsystem includes a control component, the control must additionally be reachable from the estimated user position and connected through a path of at most six hops, containing an appropriate moving joint, to as many as three associated output components. The slightly longer control path accommodates mechanisms in which user input is transmitted through intermediate linkage before reaching the functional output.

Component and structural requirements. Component presence receives score 1 when a generated role matches the required component, including accepted lexical relations and components rigidly fixed to the corresponding matched part, and 0 otherwise. For a required relationship between two components, a structural path receives full credit when an appropriate path exists through a moving joint whose moving set contains the required endpoint component. When the required components are structurally connected but the expected articulation is absent, the relationship receives partial credit of $1/2$. If no valid structural path exists, the score is 0. This distinction separates structural presence from functional articulation: connectivity alone provides evidence of partial realization, but full credit requires the motion mechanism needed by the capability.

Human-use geometry. Human-use terms are treated as standardized geometric proxies for practical interaction rather than as universal ergonomic standards. Their purpose is to provide consistent benchmark-wide tests of whether controls and usable regions occupy plausible interaction spaces. For reachability, let d denote the Euclidean distance between the relevant control and the estimated eye position. For seated objects, the eye point is placed 0.60 m above the seat top; otherwise, a standing eye point is placed 1.40 m above ground. We define

$$s_{\text{reach}} = \max\left(0, 1 - \frac{\max(0, d - 0.70)}{1.4}\right).$$

Controls within 0.70 m therefore receive full credit, after which the score decreases continuously with distance rather than introducing a brittle binary reachability boundary. Clearance is evaluated using task-specific median human occupancy boxes sampled on a 10^3 point grid. Let f_{free} denote the fraction of the corresponding occupancy volume that remains unobstructed. We define

$$s_{\text{clear}} = \min\left(1, \frac{f_{\text{free}}}{0.9}\right).$$

Thus, clearance is evaluated volumetrically rather than through isolated collision tests. A component expected to be graspable receives full grip credit when its two smallest principal extents lie between 15 mm and 70 mm. A required usable surface receives full credit when its area is at least

$$0.05\ \text{m}^2$$

and, where a task-specific expected height interval is available, the surface lies within the corresponding band. For components expected to be spatially co-located, let g denote their geometric gap. We define

$$s_{\text{coloc}} = \text{clip}\left(1 - \frac{g - 0.10}{0.20}, 0, 1\right).$$

Because these human-use rules are intentionally general geometric proxies, we avoid imposing them when they are inconsistent with the task itself. A human-geometry link is disabled whenever the corresponding benchmark reference receives a score below 0.5 on that criterion. For tasks without reference-based human calibration, such links are disabled except for ground contact. This reference gating prevents a generic anthropometric heuristic from penalizing generated designs for failing a condition that is not satisfied by the corresponding real reference.

Dynamic motion feasibility. Motion capabilities are evaluated through rigid-body simulation in MuJoCo. The simulation is intended as a standardized test of geometric and articulated feasibility rather than a high-fidelity reconstruction of material-specific real-world dynamics. All systems are evaluated under the same simulation assumptions.

Simulation uses a 1 ms timestep and the implicit-fast integrator. Parts connected by physical contact or fixed joints are grouped into rigid bodies and represented using convex-hull collision geometry. A grounded wheel attached through an approximately horizontal and centered hinge is instead represented using cylindrical collision geometry to better approximate rolling behavior. Rigid-body masses are constructed from the S.3 geometric volumes using the common density

$$1000\ \text{kg/m}^3,$$

and friction coefficients are fixed to

$$(1,\ 0.02,\ 0.002).$$

The uniform density is used as a standardized inertial proxy rather than an estimate of the true material density of each generated component. This isolates motion feasibility from errors in material prediction and provides comparable simulation conditions across generated artifacts. Before commanded actuation, the completed artifact is allowed to settle for 0.6 s. If the object tilts by at least 20° or translates by at least 0.3 times its object diagonal during this period, all motion capabilities receive zero. Such motion indicates that the completed artifact is insufficiently stable for subsequent actuation to provide a meaningful test of intended functionality.

For articulated motion claims, candidate joints are driven by a proportional-derivative servo toward their declared limits. We use the standardized mass- and scale-dependent proportional gain

$$k_p = 100MgL,$$

where M and L denote the moving mass and characteristic length, respectively. This scaling provides comparable actuation strength across structures of different sizes; it is used as a simulation control heuristic rather than as a model of a specific physical actuator. When a generated

joint does not provide an explicit motion limit, the target is defined as $\pm 90^\circ$ for a revolute joint, $\pm 360^\circ$ for a continuous joint, or $\pm$ half of the corresponding moving-part extent for a prismatic joint. Actuation uses a 1.5 s ramp followed by a 0.4 s holding period.
Let Δq denote the achieved generalized displacement and q_{target} the commanded displacement. We define

$$s_{\text{motion}} = \min\left(1, \frac{|\Delta q|}{|q_{\text{target}}|}\right) \mathbf{1}[\text{tilt} < 15^\circ].$$

The displacement ratio measures how much of the commanded motion is physically realized. The additional tilt condition prevents a mechanism from receiving motion credit when the apparent displacement is achieved by destabilizing the entire artifact rather than by successful articulation. The actuation stability threshold is intentionally stricter than the initial settling threshold because successful operation requires the artifact to remain stable while the mechanism is being used. The better of the two actuation directions is retained. For each claimed motion, we consider up to the first four candidate joints on the structural path between the moving component and the corresponding relative component.

Rolling capabilities. Rolling capabilities associated with wheels, casters, or tracks are tested separately because the existence of a rotational joint alone does not establish that the completed object can translate through rolling. The corresponding wheel joints are left free and a horizontal force equal to 10% of the object's weight is applied for 1.5 s. If d_{travel} denotes the resulting translation, rolling receives

$$s_{\text{roll}} = \min\left(1, \frac{d_{\text{travel}}}{0.276\ \text{m}}\right) \mathbf{1}[\text{tilt} < 15^\circ].$$

The fixed forcing and normalization conditions provide a common benchmark-wide test of whether the assembled wheel or track mechanism can produce meaningful translation without destabilizing the object.

Capability aggregation. Each task-level capability c consists of a set of active requirement links $\mathcal{L}_c$. Because these links represent jointly necessary conditions for the capability to be usable, capability performance is defined by its weakest active requirement:

$$w_c = \min_{\ell \in \mathcal{L}_c} s_\ell.$$

The weakest-link formulation is intentional. Averaging the requirements could assign substantial credit to a capability whose components are present but disconnected, whose mechanism cannot move, or whose control cannot be reached. In each of these cases, the capability as a whole is not operational despite success on its remaining conditions. Let $\mathcal{C}$ denote the set of applicable task-level capabilities. The final Operability score is

$$R_{4.3} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} w_c.$$

R.3 therefore evaluates functionality compositionally rather than through isolated part or joint checks. It asks whether the generated artifact integrates the required components, structural relationships, articulated mechanisms, and human-use geometry into capabilities that can plausibly function as intended. The geometric and simulation thresholds used in these tests are fixed benchmark-wide evaluation parameters and are applied identically to all systems; they are intended as consistent comparative proxies for operability rather than claims of exact real-world physical behavior.

A.5 Limitations and Future Work

Evaluation scope and interpretation. LMBuild is designed as a standardized framework for comparing generated structures, rather than as a certification procedure for real-world engineering readiness. Its metrics operationalize buildability and functionality through reproducible geometric, semantic, kinematic, and simulation-based tests. These tests provide substantially stronger evidence

than visual plausibility alone, but they do not establish that a generated artifact can be fabricated, assembled, or operated safely in every physical setting. In particular, the current evaluation does not model manufacturing tolerances, fastener strength, elastic deformation, fracture, fatigue, tool access, wiring or fluid routing, actuator limits, or sustained loading. LMBuild scores should therefore be interpreted as measures of performance under fixed benchmark criteria and as comparative evidence of physical and functional plausibility, not as proof of deployability or safety.

Approximate physical models. Several evaluation components necessarily use simplified physical assumptions. Stability focuses on tipping; assembly feasibility uses discretized insertion paths and intermediate-support checks; and operability uses rigid-body simulation with standardized density, friction, collision geometry, and actuation parameters. These choices make evaluation deterministic and comparable across heterogeneous structures, while isolating geometric and kinematic feasibility from uncertainty in inferred physical properties. Nevertheless, different parameterizations or higher-fidelity models may change individual scores, especially for structures whose operation depends on deformable components, material-specific behavior, or complex contact. Future work should study metric sensitivity and ranking stability under alternative thresholds and simulation parameters, and validate a representative subset through physical fabrication, robotic assembly, or expert mechanical assessment. Benchmark coverage. The experiments in this paper use the 50-object LMBuild-Core split. This split is deliberately coverage-oriented: it spans multiple structural families, articulation regimes, construction paradigms, and difficulty levels rather than reproducing the empirical frequency distribution of any one source dataset. Consequently, the reported results characterize performance across the selected construction regimes but should not be interpreted as population-level estimates over all real-world objects. The 2,399-object LMBuild-Full split is intended to support broader evaluation, including category-level analysis, rare mechanisms, and more difficult product-scale assemblies. Extending evaluation to the full split and to additional domains such as compliant mechanisms, electronics, and fluid-powered systems is an important direction.

Automated evaluation and human calibration. Design Quality combines deterministic measurements with two VLM judges calibrated against ratings from four human annotators. On held-out designs, the resulting metrics preserve system-level human rankings well, while item-level correlations and human-human agreement show that individual designs remain inherently more ambiguous. The present calibration covers a limited set of objects and systems and may not capture every valid design style or decomposition strategy. In addition, Wikipedia-derived capability sheets provide an auditable source of real-world knowledge but may omit uncommon components or alternative realizations. Future versions should broaden the annotator and system pool, include domain experts where appropriate, report uncertainty for agreement statistics, and continue expanding accepted component synonyms, material alternatives, and functional realizations.

System comparability and statistical uncertainty. General-purpose agents receive the same task specification, environment semantics, tool definitions, and budgets, but provider-specific execution interfaces and modality support necessarily differ. Domain-specific systems additionally retain their native inputs and outputs and are therefore comparable only on shared applicable metrics. Unless otherwise stated, the current results use one final generation trajectory per model-task pair. Confidence intervals obtained by resampling benchmark objects quantify variation across tasks, but not within-model stochasticity across repeated generations. Future evaluation should include repeated runs where computationally feasible, standardized native tool-calling interfaces, and reporting of both task-level and run-level uncertainty.

Outlook. LMBuild represents a first step toward evaluating generated structures beyond rendered appearance. A particularly promising direction is to close the loop between digital generation and physical fabrication. Future extensions could integrate additive-manufacturing constraints—including minimum feature size, wall thickness, overhangs, support requirements, print orientation, tolerances, material choice, and printer build volume—and evaluate whether generated parts can be successfully sliced, printed, post-processed, assembled, and operated. Physical validation on a representative subset, using both monolithic 3D printing and multi-part fabrication followed by human or robotic assembly, would enable direct measurement of print success, dimensional accuracy, assembly time, structural integrity, and functional performance. Such experiments could also support sim-to-real calibration of LMBuild's geometric, assembly, material, and operability metrics.

Beyond 3D printing, future work should incorporate additional manufacturing processes, richer contact and material mechanics, tool-access constraints, fasteners, compliant components, and electrical

or fluid routing. Expanding evaluation to LMBuild-Full, conducting repeated generations, and analyzing sensitivity to evaluator parameters would further improve coverage and statistical reliability. By releasing the benchmark data, capability annotations, model trajectories, fabrication specifications, and evaluation implementation, we hope to enable the community to audit these assumptions, reproduce both digital and physical evaluations, and progressively align standardized benchmark scores with manufacturability, assembly feasibility, and real-world functional performance.