VISTA-BRICK: VISUALLY FAITHFUL AND PHYSICALLY STABLE BRICK PROGRAM GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reproducing an object's geometry does not guarantee a physically stable assembly. We present VISTA-Brick, a framework for generating visually faithful and physically stable brick programs from multi-view RGB-D observations and text. To support this task, we construct a multimodal extension of StableText2Brick that pairs source-object observations with reference brick programs. VISTA-Brick combines multimodal supervised fine-tuning with online GRPO, using program-level geometric agreement, structural cues, and static-analysis feedback to evaluate the policy's own predictions. During post-training, the physical solver serves as an evaluator rather than repairing sampled programs. We evaluate the learned policy both through direct generation and with optional budgeted physical search. Compared with the supervised baseline, post-training improves program legality and static stability without inference-time correction, while modestly improving geometric fidelity. Under the same physical search procedure and sampling-budget cap, post-trained policies also achieve higher joint geometric–physical success with lower average search cost. These results support learning structural feasibility alongside geometric fidelity, with physical verification complementing rather than substituting for the learned generation capability.

1 INTRODUCTION

Generating a 3D shape that resembles a target is not the same as finding a physically feasible way to construct it. In brick generation, an output must specify not only an object's geometry, but also its decomposition into discrete components and their spatial arrangement. Two assemblies can occupy the same volume while differing in their connections and structural stability (Xu et al., 2026). Conversely, satisfying physical constraints alone does not ensure that an assembly preserves the intended object's geometry (Yuan et al., 2026). Brick program generation therefore provides a concrete setting for studying a broader challenge: how to translate visual evidence into structured outputs that satisfy both instance-level geometric requirements and verifiable physical constraints.

Recent work has made progress along several complementary directions. BrickGPT (Pun et al., 2025) generates brick sequences from text and combines autoregressive prediction with validity checks and physics-aware rollback. Inference-time verification can detect physical failures, but repeated rejection and suffix regeneration consume additional computation. Image-conditioned methods such as LEGO-Maker (Ge et al., 2025) introduce visual references to guide brick generation. Post-training methods further incorporate geometric and structural feedback: PVPO (Yuan et al., 2026) couples physical-validity and voxel-alignment rewards for text-conditioned synthesis, while STABLE Xu et al. (2026) studies rollback-free generation from complete occupied-voxel coordinates derived from target brick structures. These approaches address different aspects of conditional generation and physical feasibility. Our focus is on their interaction in an observation-conditioned setting, where the goal is to obtain a geometrically faithful, physically verified assembly within a limited search budget. This motivates our central question: Can program-level visual and physical feedback make a multimodal generator more likely to produce instance-faithful, stable brick assemblies, while reducing the search needed to obtain them?

So we present **VISTA-Brick**, a framework for *visually faithful and physically stable brick program generation* from multi-view RGB-D observations and text. VISTA-Brick combines two-stage policy training with explicit physical verification at inference. In the first stage, we perform su-

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.

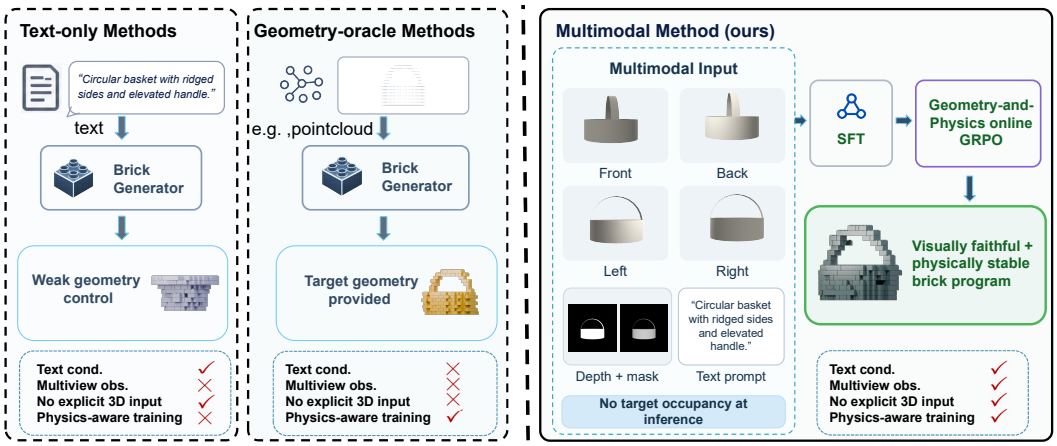

Figure 1: Overview of VISTA-Brick and alternative conditioning paradigms. **Left:** Text-only methods generate from descriptions that may leave instance-specific geometry underspecified. **Middle:** Geometry-oracle methods receive target geometry directly as input. **Right:** Multimodal supervised fine-tuning establishes the observation-to-program mapping, followed by online GRPO with program-level geometric and physical feedback. Reference brick geometry is not provided as an inference input.

pervised fine-tuning of Qwen3-VL-2B (Bai et al., 2025) on a multimodal corpus constructed from ShapeNet meshes (Chang et al., 2015). In the second stage, we apply online Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to candidates sampled from the current multimodal policy. Each program receives a composite geometric score and physical score. A reward penalizes illegal programs and candidates with low geometric fidelity. The physical solver only evaluates training candidates; it does not repair them through rollback during policy optimization. At inference, VISTA-Brick supports direct generation and optional budgeted physical search. Following BrickGPT (Pun et al., 2025), the latter combines local validity constraints with Gurobi-based static analysis (Gurobi Optimization, LLC, 2026), using stable-prefix rollback and suffix regeneration when verification fails.

Our contributions are threefold:

- **A multi-view RGB-D data resource.** We construct a multimodal extension of StableText2Brick (Pun et al., 2025), pairing its reference brick programs and captions with four-view RGB images, depth maps, and foreground masks. The resulting resource supports observation-conditioned brick program learning.
- **Observation-conditioned brick program generation.** We develop VISTA-Brick, a framework that learns to generate brick programs from multi-view RGB-D observations and text. It combines multimodal supervised learning with explicit geometric and static-stability verification to search for instance-faithful, physically feasible assemblies.
- **Program-level geometric-physical post-training.** We apply online GRPO to uncorrected programs sampled from the observation-conditioned policy, combining reference-based geometric agreement with structural and static-analysis feedback. During evaluation, post-training improves direct generation and increases joint geometric-physical success under an unchanged physical search procedure, while reducing average sampled-token consumption.

2 Related Work

Conditional brick generation. Brick generation uses language or geometric observations to specify the intended object. Visual and multimodal approaches learn brick sequences or stepwise placement decisions from image observations (Chung et al., 2021; Guo et al., 2024; Ge et al., 2025;

Xu et al., 2025; Liu et al., 2026), providing instance-specific evidence beyond text descriptions. A complementary route constructs assemblies from explicit geometry, either supplied directly or reconstructed from observations (Ni et al., 2026; Chen et al., 2026). The information available to the generator is an important distinction: STABLE (Xu et al., 2026), for example, receives complete occupied-voxel coordinates of the target brick structure. VISTA-Brick instead learns directly from multi-view RGB-D observations and text, without requiring a completed mesh or target occupancy as its inference input.

Programmatic shape and assembly representations. Beyond explicit surfaces and occupancy, programmatic approaches describe shapes through constructive operations or part-assembly instructions (Sharma et al., 2018; Jones et al., 2020; Wu et al., 2021). These representations expose how a shape is constructed, supporting structured generation and editing. Brick-specific methods similarly encode assemblies through component types and poses or explicit inter-brick relations (Pun et al., 2025; Xu et al., 2025; Ni et al., 2026; Kulits & Schmid, 2026). Relational representations make assembly dependencies explicit; for example, BrickNet (Kulits & Schmid, 2026) parameterizes assemblies through connector relationships, while attachment-tree representations organize local placement relations (Guo et al., 2024; Ni et al., 2026). VISTA-Brick retains coordinate-based programs and focuses on learning their geometric and structural quality from instance observations and program-level feedback.

Physical learning and verification. Physical feasibility has been addressed through geometric refinement and constraint-aware assembly (Prévost et al., 2013; Luo et al., 2015; Ahn et al., 2022). More recent approaches incorporate geometric and structural objectives into reinforcement learning (Xu et al., 2026; Yuan et al., 2026) or preference-based post-training using DPO (Rafailov et al., 2023; Xu et al., 2025; Ni et al., 2026). In particular, PVPO (Yuan et al., 2026) couples physical-validity and target-voxel rewards to address the mismatch between valid assemblies and the intended geometry. Inference-time verification offers a complementary mechanism: BrickGPT (Pun et al., 2025) combines local validity checks with physics-aware suffix rollback. VISTA-Brick applies online GRPO (Shao et al., 2024) to raw programs sampled from an observation-conditioned policy, using geometric and structural–static feedback with legality and geometric gating. Its direct and verifier-assisted inference modes allow us to examine how learned feasibility complements explicit physical search.

3 DATASET

We construct a multi-view RGB-D extension of StableText2Brick (Pun et al., 2025) for observation-conditioned brick program generation. StableText2Brick provides captions and reference brick programs, while our extension adds synchronized appearance and geometric observations rendered from the corresponding source-object meshes. Importantly, the images are not rendered from the target brick assemblies. The learning task is to translate object observations into a discrete brick realization.

3.1 MULTIMODAL PAIR CONSTRUCTION

We associate each brick structure with its ShapeNetCore mesh (Chang et al., 2015) using the category and object identifiers inherited from the source data. A separate structure identifier distinguishes multiple brick realizations associated with the same source object. We retain the five captions and reference program provided for each structure.

Each multimodal record is represented as

$$(\mathcal{O}, c, B^\star), \qquad \mathcal{O} = \{I_v, D_v, M_v\}_{v\in\mathcal{V}}, \quad \mathcal{V} = \{\text{front}, \text{right}, \text{back}, \text{left}\}, \tag{1}$$

where c is a caption, $B^\star$ is the reference brick program, and I_v, D_v, and M_v denote the RGB image, normalized depth map, and foreground mask for view v. Since the brick program is a discrete approximation of the source object, the source mesh and the reference assembly are not assumed to be pointwise identical. More details on the rendering and model inputs are provided in Appendix A.2.

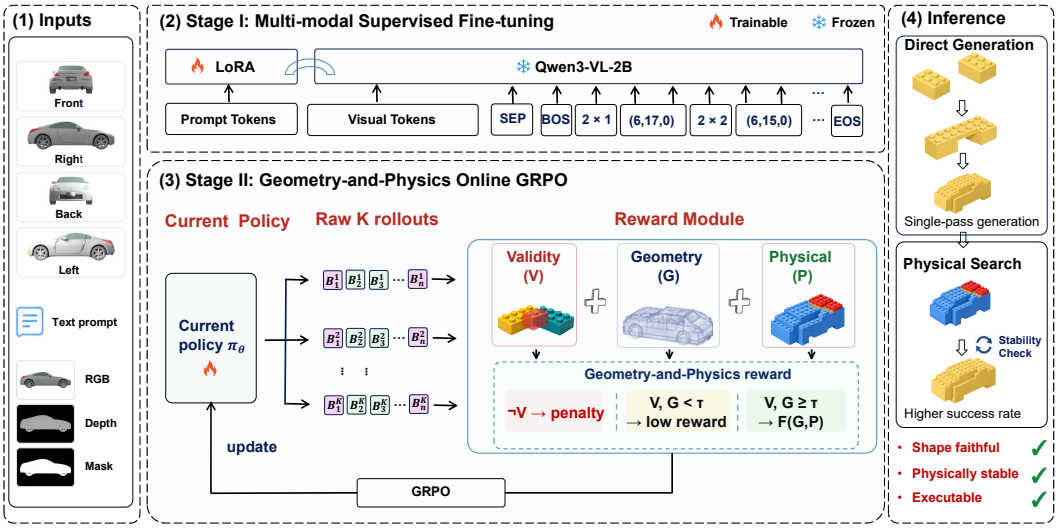

Figure 2: Overview of VISTA-Brick. Multimodal supervised fine-tuning first learns the mapping from multi-view RGB-D observations and text to brick programs. Online GRPO then evaluates raw programs sampled from the current policy using geometric agreement, structural cues, and static-analysis feedback.

3.2 Dataset scale and splits

The resulting resource covers 21 object categories. The training partition contains 42,604 reference brick programs associated with 25,653 source objects. Expanding the five captions of each structure produces 213,020 caption–program training records. The source evaluation partition contains 250 programs associated with 250 objects, yielding 250 records.

Caption expansion increases the number of language conditions but does not create additional geometries or rendered view sets. We preserve the original train–test partition before caption expansion, and the two partitions have no overlapping object keys or structure identifiers. For supervised learning, all caption-expanded records are used. Dataset construction is indexed by explicit view names and checks the availability of the required RGB, depth, and mask files before a record is included.

4 Method

VISTA-Brick generates brick programs from multi-view RGB-D observations and text. As illustrated in Figure 2, training consists of two stages. Supervised fine-tuning learns an observation-to-program policy, after which online policy optimization evaluates the model's own generated programs using geometric and structural feedback. During post-training, the physical solver acts only as a non-differentiable evaluator and does not repair sampled programs. At inference, the learned policy can be used directly or combined with an optional verification-and-regeneration procedure.

4.1 Problem formulation and multimodal policy

Let $\mathcal{V} = \{\text{front}, \text{right}, \text{back}, \text{left}\}$ denote the four canonical views. For each view v, the input contains an RGB image I_v, a normalized depth image D_v, and a foreground mask M_v. We encode depth and foreground information as a separate three-channel image rather than adding a fourth channel to RGB. Defining $d_v = D_v/255$, $m_v = M_v/255$, and $a_v = \mathbf{1}[M_v > 127]$, we construct

$$E_v = [d_v,\ m_v,\ (1 - d_v) \odot a_v]\,. \tag{2}$$

Each RGB–depth pair is preceded by its textual view label and provided to the native visual interface of Qwen3-VL-2B-Instruct (Bai et al., 2025). The resulting conditioning input is

$$X = (c, \{I_v, E_v\}_{v \in \mathcal{V}}), \tag{3}$$

where c is the text description.

The target is a variable-length brick program $B = (b_1, \ldots, b_N)$ in a $20 \times 20 \times 20$ workspace. Each brick is represented as

$$b_i = (s_i^x, s_i^y, x_i, y_i, z_i), \tag{4}$$

where s_i^x and s_i^y are its planar dimensions and (x_i, y_i, z_i) is the minimum occupied grid coordinate. All bricks occupy one vertical grid layer, and a horizontal quarter-turn is represented by exchanging the two planar dimensions.

A serialization function maps B to a text-token sequence $Y = (y_1, \ldots, y_T)$ using the backbone tokenizer:

$$\pi_\theta(Y \mid X) = \prod_{t=1}^{T} \pi_\theta(y_t \mid X, y_{<t}). \tag{5}$$

We distinguish content validity from sequence completion. A returned program is content-valid if its parsed bricks use supported dimensions, remain inside the workspace, and do not overlap. End-of-sequence completion is tracked separately. Content validity is also distinct from global connectivity and static stability.

4.2 Supervised observation-to-program learning

We first adapt the pretrained vision–language model using LoRA (Hu et al., 2021), while keeping the backbone weights frozen. Given a reference program $Y^\star$, teacher-forced training minimizes the response-token cross-entropy

$$\mathcal{L}_{\mathrm{SFT}} = -\frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \log \pi_\theta \left(y_t^\star \mid X, y_{<t}^\star\right), \tag{6}$$

where $\mathcal{T}$ contains the target response positions; prompt, image-placeholder, and padding positions are excluded.

Training mixes multimodal caption conditioning, multimodal generic instructions, and caption-only conditioning, all supervised by the same reference program. We also randomly drop paired RGB and depth views as an observation-missing augmentation. Exact sampling and adapter configurations are provided in Appendix C. No physical solver or rollback is used during supervised fine-tuning.

4.3 Online Policy Optimization with Geometric and Physical Feedback

Teacher-forced supervision does not directly assess the complete programs produced by the model itself. We therefore initialize an online policy from the supervised checkpoint and retain a frozen copy as the reference policy π_{ref}. For each condition X, the rollout policy π_{old} samples a group of raw responses $\{Y_i\}_{i=1}^{K}$. Each response is processed by the reward parser, yielding the brick list $\widehat{B}_i$ used for scoring and a content-validity indicator $v_i \in \{0, 1\}$. The treatment of malformed and truncated responses is specified in Appendix B.

Geometric agreement. We compare $\widehat{B}_i$ with its paired reference program $B^\star$ in their shared brick coordinate system:

$$G_i = 0.40 I_{\mathrm{vox}} + 0.35 I_{\mathrm{sil}} + 0.25 S_{\mathrm{depth}}. \tag{7}$$

I_{vox} is voxel intersection-over-union, I_{sil} is the mean silhouette IoU over the two orthogonal horizontal projection axes, and S_{depth} averages depth agreement over four horizontal views. All components are normalized to $[0, 1]$. The paired reference program is used only for training and evaluation, and is not available at inference.

Structural and static quality. For content-valid candidates, we compute

$$P_i = 0.30S_{\text{static}} + 0.25C_{\text{ground}} + 0.25I_{\text{support}} + 0.20J_{\text{offset}}. \tag{8}$$

Here, S_{static} is a normalized score derived from the BrickGPT static analyzer (Pun et al., 2025); C_{ground} is the fraction of bricks reachable from ground bricks through adjacent-layer contacts; I_{support} is the fraction of non-ground bricks with multiple lower-layer supports; and J_{offset} measures the prevalence of offset seams across adjacent layers. The latter three terms are structural proxies and do not independently certify static stability. The exact solver-to-score mapping is provided in Appendix B.

Program-level reward. We use a piecewise reward to encode an explicit preference over program legality, minimum geometric agreement, and joint geometric-physical quality. Physical quality contributes to the reward only for content-valid candidates whose geometric score reaches a prescribed threshold.

$$R_i = \begin{cases} -1 + 0.10G_i, & v_i = 0, \\ -0.50 + G_i, & v_i = 1 \wedge G_i < \tau, \\ 0.10 + 0.90\left(\lambda G_i + (1-\lambda)P_i\right), & v_i = 1 \wedge G_i \geq \tau, \end{cases} \tag{9}$$

where $\tau = 0.40$ and $\lambda = 0.70$. The reward assigns legal candidates that pass the geometric threshold a higher score than legal candidates below it, irrespective of their physical score. We use this piecewise formulation as one implementation of joint visual–physical feedback. Our focus is on whether combining geometric and physical feedback improves the observation-conditioned policy through post-training, rather than on prescribing a particular reward aggregation scheme.

Group-relative update. For candidates generated under the same condition, we normalize their rewards within the group:

$$A_i = \frac{R_i - \overline{R}}{\max(\sigma_R, \delta)}, \qquad \overline{R} = \frac{1}{K}\sum_{j=1}^{K} R_j, \tag{10}$$

where σ_R is the within-group standard deviation and δ is a numerical floor. The same program-level advantage is assigned to all completion tokens.

Following GRPO (Shao et al., 2024), we optimize

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{T_i}\sum_{t=1}^{T_i}\Big[-\min\left(\rho_{it}A_i, \text{clip}(\rho_{it}, 1-\epsilon, 1+\epsilon)A_i\right) + \beta k_{it}\Big], \tag{11}$$

where

$$\rho_{it} = \frac{\pi_\theta(y_{it} \mid X, y_{i,<t})}{\pi_{\text{old}}(y_{it} \mid X, y_{i,<t})}$$

and k_{it} is the sampled-token KL estimator with respect to the frozen supervised reference. The rollout policy is refreshed after each online update round.

4.4 Direct and verifier-assisted inference

VISTA-Brick supports two inference modes. In direct generation, the learned policy emits a complete program without external rejection or repair, allowing its unassisted generation capability to be evaluated. In verifier-assisted generation, we follow the constrained decoding and rollback paradigm of BrickGPT (Pun et al., 2025). Syntax constraints are applied during decoding, and each proposed brick is checked for supported dimensions, workspace bounds, and collisions. Illegal placements are rejected without discarding the accepted prefix. Once a sequence terminates, the complete assembly is evaluated by the static analyzer. A failed candidate triggers rollback to a verifier-determined prefix, followed by suffix regeneration until a feasible candidate is found or the search budget is exhausted. Direct generation and verifier-assisted generation are evaluated separately, distinguishing improvements in the learned policy from improvements obtained through external search.

Table 1: Results on the 250-object evaluation set. Bold identifies the best displayed value within each panel; identical rounded values are tied.

(a) Direct generation without static-physics rollback

Method	Valid ↑	Stable ↑	Visual ↑	Voxel IoU ↑	Joint@0.6 ↑
BrickGPT (no physical rollback)	**100.0**	28.8	0.4081	0.2749	8.4
STABLE	93.6	58.0	0.6454	0.4940	41.6
TreeSBA	**100.0**	**76.0**	0.4300	0.2612	13.2
VISTA-Brick (SFT)	64.4	44.4	0.7759	0.6489	42.8
VISTA-Brick (GRPO)	86.4	64.8	**0.7916**	**0.6733**	**63.2**

(b) Physically assisted generation

Method	Stable ↑	Visual ↑	Voxel IoU ↑	Joint@0.6 ↑	Tokens ↓
BrickGPT + rollback	86.0	0.4045	0.2725	17.6	8.55k
Hunyuan3D-2mv + Mesh2Brick	70.8	0.6899	0.5200	51.6	–
VISTA-Brick (SFT) + rollback	84.4	0.7648	0.6317	79.2	7.56k
VISTA-Brick (GRPO) + rollback	**94.0**	**0.7648**	**0.6367**	**87.2**	**5.02k**

5 EXPERIMENTS

We evaluate VISTA-Brick and the baselines on 250 objects across different categories. We consider two protocols: *direct generation*, which produces one greedy program without constraints, rejection, or rollback; and *physical search*, which combines constrained sampling, static verification, and rollback under a 32K cumulative sampled-token budget, including rejected and discarded tokens. We report program legality (*Valid*), static-verification success (*Stable*), geometric agreement (*Geometry*; equation 7), voxel IoU, and *Joint@0.6*, which requires both stability and Visual ≥ 0.6, alongside average token consumption and search time. All success rates retain failed objects in the denominator. Detailed protocols, metric definitions, failure handling, and implementation settings are provided in the appendix D ,appendix B, and appendix C. For the reconstruction-based baseline, we use the multi-view checkpoint from the Hunyuan3D 2.0 family (Zhao et al., 2025) for mesh reconstruction, followed by physics-assisted conversion. More details are given in Appendix D.3.

5.1 COMPARISONS

Post-training improves unassisted generation. Table 1(a) evaluates one greedy output per object without rejection, repair, or physical rollback. Compared with SFT, GRPO-10K improves validity from 64.4% to 86.4%, stability from 44.4% to 64.8%, and Joint@0.6 from 42.8% to 63.2%. Geometric quality also improves, with Visual increasing from 0.7759 to 0.7916 and voxel IoU from 0.6489 to 0.6733. These results show that program-level post-training improves the policy's ability to produce feasible structures directly, rather than relying on inference-time correction.

BrickGPT achieves high program validity, while the adapted TreeSBA additionally obtains the highest direct stability rate. STABLE maintains high validity and achieves stronger geometric agreement than text-only BrickGPT using an oracle point cloud sampled from the ground-truth mesh. However, these individual strengths do not consistently translate into assemblies that are both stable and faithful to the target, as reflected in their lower Joint@0.6 scores. In contrast, VISTA-Brick achieves the highest geometric scores and joint success rate among the evaluated direct-generation methods. Its advantage lies not in maximizing validity or stability alone, but in more frequently producing physically stable assemblies that also preserve the geometry of the observed instance.

Post-training improves verified success and search efficiency. Table 1(b) and Figure 3 report results with physical assistance. Under the same rollback procedure and 32K sampled-token cap, VISTA-Brick increases stability from 84.4% to 94.0% and Joint@0.6 from 79.2% to 87.2% relative to SFT. At the 32K budget, mean Visual remains unchanged at the reported precision, while voxel

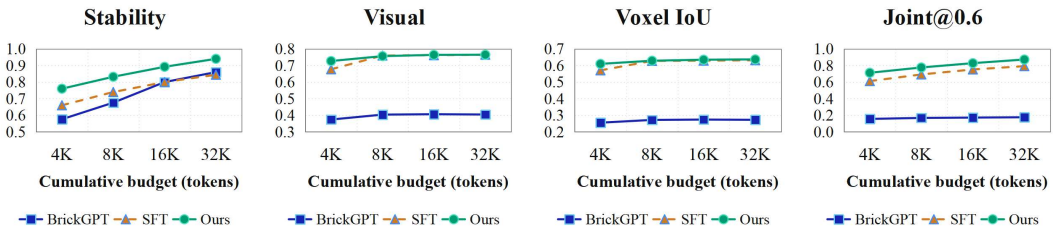

Figure 3: Verified generation performance under different sampled-token budgets. Under the same physical search procedure, the post-trained policy achieves higher static-verification success than RGB-D SFT across the evaluated sampled-token budgets, with comparable aggregate geometric agreement. Budget curves are reconstructed from saved search trajectories.

IoU increases from 0.6317 to 0.6367. The post-trained policy therefore provides better starting candidates for physical search while preserving the geometric quality learned during SFT.

VISTA-Brick also reduces mean sampled-token consumption from 7.56k to 5.02k, a 33.6% reduction per object. The main benefit is thus not only a higher final stability rate, but also a reduction in the amount of regeneration required to obtain a verified structure. This supports a complementary relationship between learning and verification: post-training improves the candidate distribution, while physical search resolves the remaining failures.

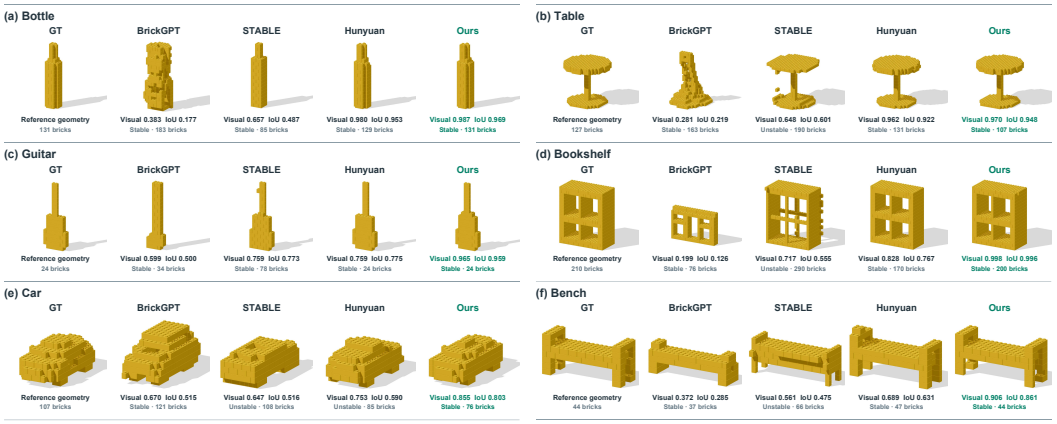

Figure 4: Qualitative comparison across six object categories. GT denotes the reference brick geometry. Each row uses the same camera and scale across methods.

Qualitative comparison. Figure 4 compares selected outputs across six object categories against the reference brick geometry. VISTA-Brick closely preserves instance-specific proportions and part layouts, including the guitar's neck-to-body ratio, the table's slender central support, and the bookshelf's internal compartments. BrickGPT produces stable assemblies in these examples, but its bookshelf and guitar differ substantially from the references in scale and proportions. STABLE captures several major shape features, yet its table contains detached pieces and its bookshelf changes the internal shelf layout; both outputs are marked unstable. Hunyuan closely reconstructs the round table, whereas its bookshelf reconstruction shows noticeable deviations. The selected VISTA-Brick outputs combine close geometric agreement with verified stability, illustrating the distinction between generating a plausible object and recovering a physically feasible realization of the target.

5.2 Ablation Study

We examine visual conditioning and reward design on the evaluation dataset.

Table 2: Visual observation ablation on 250 objects using the same GRPO checkpoint. Physical search uses a 32K cumulative sampled-token budget. Valid, Stable, and Joint@0.6 are percentages. Bold denotes the best value within each inference setting.

Inference	Input	Valid ↑	Stable ↑	Visual ↑	Voxel IoU ↑	Joint@0.6 ↑
Direct	Text-only	**93.6**	**84.0**	0.4578	0.3368	21.6
Direct	Text+RGB-D	86.4	64.8	**0.7916**	**0.6733**	**63.2**
+Search	Text-only	99.2	90.0	0.4380	0.3126	23.2
+Search	Text+RGB-D	**100.0**	**94.0**	**0.7648**	**0.6367**	**87.2**

Table 3: Reward ablation at matched training steps on 250 objects. All GRPO variants are evaluated after the same training steps, with SFT as their starting baseline. Physical search uses a 32K cumulative sampled-token budget. Valid, Stable, and Joint@0.6 are percentages. Bold denotes the best value within each inference setting.

Inference	Method	Valid ↑	Stable ↑	Visual ↑	Voxel IoU ↑	Joint@0.6 ↑
Direct	SFT	64.4	44.4	0.7759	0.6489	42.8
Direct	Geometry-only GRPO	72.0	47.2	0.7693	0.6428	45.6
Direct	Full GRPO	**73.6**	**49.6**	**0.7796**	**0.6542**	**48.8**
+Search	SFT	99.6	84.4	0.7648	0.6317	79.2
+Search	Geometry-only GRPO	**100.0**	86.4	0.7669	0.6367	81.6
+Search	Full GRPO	**100.0**	**87.6**	**0.7685**	**0.6372**	**82.4**

Visual conditioning. We compare text-only and text+RGB-D inputs using fixed policy weights and identical text descriptions, with decoding settings held constant within each protocol. Table 2 shows that adding RGB-D substantially improves geometric agreement and increases direct Joint@0.6 from 21.6% to 63.2%, despite lower stability when considered alone. With physical search, both conditions achieve high stability, but their joint success rates remain widely separated: 23.2% for text-only versus 87.2% for text+RGB-D. These results highlight the role of visual observations in specifying instance geometry: physical verification can improve feasibility, but does not replace the geometric information supplied by the observations.

Reward design. Starting from the same SFT checkpoint, we compare geometry-only GRPO with the full visual–physical configuration after the same training steps, using identical multimodal inputs and direct greedy evaluation. As shown in Table 3, both variants improve validity, stability, and joint success over SFT. The full configuration further improves stability from 47.2% to 49.6% and Joint@0.6 from 45.6% to 48.8% relative to geometry-only GRPO, while also achieving higher geometric scores. Only the full configuration exceeds SFT on both Visual and voxel IoU in these runs, supporting the combined objective for improving structural quality alongside geometric fidelity.

6 Conclusion

We presented VISTA-Brick, a framework for generating visually faithful and physically stable brick programs from multi-view RGB-D observations and text. Our multimodal extension of StableText2Brick pairs source-object observations with reference programs to support instance-conditioned learning. VISTA-Brick incorporates program-level geometric and physical feedback into the observation-conditioned policy. On the evaluation dataset, this post-training improves program legality, static-verification success, and joint geometric-physical success without inference-time correction. Overall, VISTA-Brick supports a complementary role for learning and verification: program-level feedback improves the policy's own generation capability, while explicit verification helps turn its proposals into assemblies that satisfy the adopted physical constraints.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and polishing, as well as the retrieval and discovery of relevant literature. Specifically, these tools helped improve the grammar, clarity, and readability of author-written text and identify potentially relevant prior work. All AI-assisted edits were reviewed and revised by the authors to ensure that they accurately reflected the intended meaning and technical content. References suggested by AI tools were checked against the original sources for accuracy and relevance before inclusion in the paper. We take full responsibility for the final content of this work, including all text, claims, and references produced with the assistance of generative AI tools.

REPRODUCIBILITY STATEMENT

We support reproducibility through the methodological descriptions, experimental protocols, and accompanying source code. The method section describes the multimodal inputs, brick-program representation, supervised fine-tuning procedure, and online GRPO objective, including the geometric and physical reward components. The experiments section and appendix provide the dataset construction procedure, training settings, baseline configurations, and evaluation protocols for direct generation and budgeted physical search, together with the metric definitions and token-budget accounting rules. Core source code for multi-view data construction, multimodal SFT, online GRPO, and physical stability analysis, along with training configurations, is available for anonymous review at `https://anonymous.4open.science/r/vista-brick-review-0F54/`.

REFERENCES

Seokjun Ahn, Jungtaek Kim, Minsu Cho, and Jaesik Park. Budget-aware sequential brick assembly with efficient constraint satisfaction. *arXiv preprint arXiv:2210.01021*, 2022.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Xilun Chen, Hanwen Wan, Yusong Zhao, Zexin Lin, Ruixiang Liao, and Xiaoqiang Ji. Resembrick: Brick reconstruction from photographs with perceptual fidelity and buildability. *arXiv preprint arXiv:2608.09597*, 2026.

Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W Taylor, Jaesik Park, and Minsu Cho. Brick-by-brick: Combinatorial construction with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5745–5757, 2021.

Jiahao Ge, Mingjun Zhou, Hanyou Zheng, Hao Xu, and Chi-Wing Fu. Lego®-maker: Autoregressive image-conditioned lego® model creation. *ACM Transactions on Graphics (TOG)*, 44(6): 1–15, 2025.

Mengqi Guo, Chen Li, Yuyang Zhao, and Gim Hee Lee. Treesba: Tree-transformer for self-supervised sequential brick assembly. In *European Conference on Computer Vision*, pp. 35–51. Springer, 2024.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL `https://www.gurobi.com`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Transactions on Graphics (TOG)*, 39(6):1–20, 2020.

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

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.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: Optimizing lego designs. *ACM Transactions on Graphics (ToG)*, 34(6):1–12, 2015.

Zhengyang Ni, Feng Yan, Yu Guo, and Fei Wang. Brickanything: Geometry-conditioned buildable brick generation with structure-aware tokenization. *arXiv preprint arXiv:2605.26182*, 2026.

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make it stand: balancing shapes for 3d fabrication. *ACM Transactions on Graphics (ToG)*, 32(4):1–10, 2013.

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, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5515–5523, 2018.

Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Modelnet40-c: A robustness benchmark for 3d point cloud recognition under corruption. In *ICLR 2022 workshop on socially responsible machine learning*, volume 7, 2022.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6752–6762. IEEE, 2021.

Chenhui Xu, Ziyue Bai, Fuxun Yu, Heng Huang, and Jinjun Xiong. Rollback-free stable brick structures generation. *arXiv preprint arXiv:2605.06947*, 2026.

Hao Xu, Yuqing Zhang, Yiqian Wu, Xinyang Zheng, Yutao Liu, Xiangjun Tang, Yunhan Yang, Ding Liang, Yingtian Liu, Yuanchen Guo, et al. Legoace: Autoregressive construction engine for expressive lego® assemblies. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, 2025.

Yuhuan Yuan, Zhouliang Yu, Minghao Liu, Weiyang Liu, and Ge Lin Kan. Sample-efficient post-training for lego spatial-physics reasoning. *arXiv preprint arXiv:2606.07602*, 2026.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

A Dataset Construction and Split Audit

A.1 Source records and evaluation pairing

We retain the reference brick programs, captions, and source partition of StableText2Brick, and attach observations of the corresponding ShapeNet source objects. Object identity is the pair (category ID, object ID); structure identity is the structure UUID. A source object can have several reference structures. Each structure has five caption–program records in SFT. GRPO uses one record per structure, giving 42,604 conditions rather than 213,020 caption-expanded records. The three SFT conditioning modes are sampled during collation; they do not create three additional copies of the dataset.

Table 4: Manifest counts, recomputed from the saved data files. The GRPO partition contains the same objects and structures as the SFT training partition.

Partition	Objects	Structures	Records	Categories
train	25,653	42,604	213,020	21
grpo	25,653	42,604	42,604	21
eval	250	250	250	20

The source train and evaluation partitions have zero overlapping object IDs and zero overlapping structure UUIDs. That means the evaluation set has zero overlap with SFT or GRPO training under both identifiers. With seed 20260907, quotas are allocated across categories in shuffled round-robin order. Objects are then sampled without replacement; one caption–structure record is randomly selected from each object's records, sorted by sample ID. The resulting manifest fixes the pairing for all methods. No model score or GT stability filter is used at selection. Table 10 reports the resulting distribution.

A.2 Rendering and model inputs

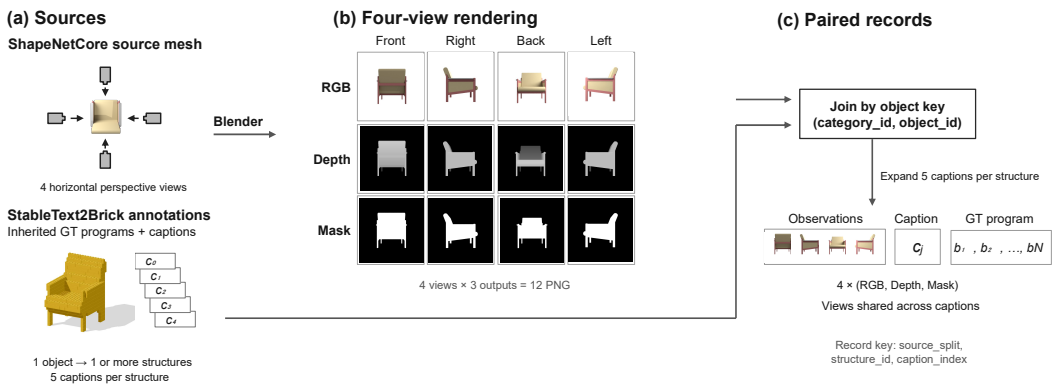

Figure 5: Construction of our multimodal brick-generation dataset. We associate each ShapeNetCore source mesh with the corresponding StableText2Brick annotations, render four canonical RGB, depth, and foreground-mask views in Blender, and pair these observations with the inherited captions and reference brick programs using object identifiers. Each structure is expanded over its five captions while sharing the same multi-view observations.

The renderer imports normalized ShapeNet OBJ files with source axes $+Y$ up and $-Z$ front, converting them to Blender's $+Z$-up frame. The camera targets the world-space bounding-box center c. Let r be the largest distance from c to a mesh bounding-box corner. Camera distance is $\ell = 1.2r/\sin(45^\circ/2)$. The four horizontal azimuths are $0^\circ, 90^\circ, 180^\circ, 270^\circ$, labeled front, right, back, and left. The saved metadata for all 28,528 objects records 512×512 images, a 45° field of view, Cycles, and 16 samples.

Textures are replaced by flat materials derived from each source material's average color, with luminance and channel limits; missing materials use gray. Thus, this rendering configuration does not assign one identical gray material to every object. A shared near–far interval is used across each object's views: $n = \max(10^{-4}, \ell - 1.05r)$ and $f = \ell + 1.05r$. The depth pass is mapped from $[n, f]$ to $[1, 0]$ and saved as an 8-bit grayscale PNG. Background depth is zero and is distinguished using a separate mask. The mask and RGB/depth images share the same camera and resolution.

For each view, the loader constructs a three-channel image from uint8 arrays:

$$E_v = \big[D_v,\ M_v,\ (255 - D_v)\,\mathbf{1}(M_v > 127)\big].$$

The VLM receives alternating RGB and encoded-depth images in the order front, right, back, left, with text labels: eight images in total. Camera poses, near/far values, object bounds, mesh vertices, and reference programs are not included in its inference prompt. The saved processor outputs for all 250 multimodal direct generations contain eight image grids of shape $(1, 32, 32)$; with spatial merging by two, each image contributes 256 visual tokens. The data files also retain camera and bounds metadata, which are not provided to the VLM. STABLE receives a point cloud sampled from the ground-truth mesh. All structures paired to a source object share its rendered observations. Because one source object may admit multiple reference brick realizations, observation-only conditioning is inherently one-to-many. Our reference-based reward and evaluation use the structure paired with each condition; alternative valid realizations are not treated as equivalent targets. Geometry scores compare generated and reference programs in their stored integer grid; evaluation does not align each prediction to its reference.

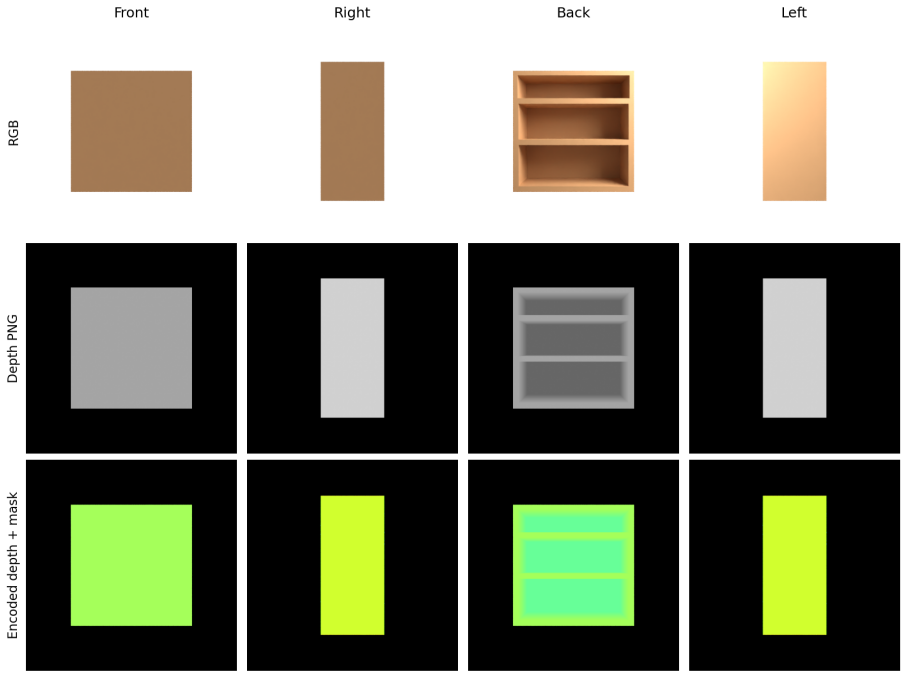

Figure 6: Recorded observations for fixed evaluation object 0. Rows show RGB, the saved depth PNG, and the actual depth–mask encoding passed to the VLM. The complete caption and reference sequence are retained with the evidence package.

B Program Parsing and Reward Specification

B.1 Program representation and parsing

A program is a newline-separated list of `HxW (x,y,z)` records. The supported oriented footprints are $(1, 1)$, $(1, 2)$, $(2, 1)$, $(2, 2)$, $(1, 4)$, $(4, 1)$, $(2, 4)$, $(4, 2)$, $(1, 6)$, $(6, 1)$, $(2, 6)$, $(6, 2)$, $(1, 8)$, and $(8, 1)$. Each brick occupies one vertical grid layer. Coordinates are nonnegative integers and the

complete footprint must lie inside $[0, 20)^3$. Programs preserve their serialized order; inference does not sort the output. Completion is determined from the model's EOS token, separately from content parsing.

The parser strips each line and ignores empty lines, then uses a full-line regular expression. Prose and code fences are not removed. It stops at the first malformed or unsupported brick and returns the preceding parsed prefix with an error flag. The scorer does *not* score that prefix as a repaired prediction: any parse error yields empty scored occupancy and invalid content. The training cap is 480 nonblank lines; final evaluation uses 2,000. The cap check occurs before line parsing.

Table 5: Parser and completion rules. Geometric agreement is not automatically zero for a parseable colliding or out-of-bounds program.

Response state	Content/geometry treatment	Completion treatment
Well-formed, in-bounds, disjoint	Valid if nonempty; score occupied cells	EOS checked separately
Prose or malformed line	Invalid; scored occupancy empty, even with an earlier prefix	May still have EOS
Repeated or intersecting bricks	Invalid; score the occupancy union	EOS checked separately
Out-of-bounds brick	Invalid; occupancy clipped to the evaluation cube	EOS checked separately
Empty or over-cap response	Invalid; empty scored occupancy	EOS alone does not make it valid
Truncated at a line boundary	Content can be valid	Without EOS, primary direct metrics fail
Truncated inside a line	Parse error; scored occupancy empty	Without EOS, primary direct metrics fail

B.2 Geometric agreement

For a parsed brick (h, w, x, y, z), occupancy consists of cells $\{(i, j, z) : x \le i < x + h,\ y \le j < y + w\}$ inside the grid. Let O and O^* be generated and reference occupancy sets. Voxel IoU is $|O \cap O^*|/|O \cup O^*|$, with value zero for an empty union. Let Π_{xz} and Π_{yz} denote orthographic binary projections. Silhouette agreement is the mean of the two projected IoUs; opposite views have identical binary silhouettes and are not counted twice.

The four depth maps store first-hit integer grid distances: front uses $\min y$ at each (x, z), back uses $19 - \max y$, left uses $\min x$ at each (y, z), and right uses $19 - \max x$. For view v, let U_v be the union of predicted and reference foreground pixels and I_v their intersection. Then

$$D_v = \frac{1}{|U_v|} \sum_{p \in I_v} \left(1 - \frac{|d_v(p) - d_v^*(p)|}{19}\right), \qquad D_v = 0 \text{ if } U_v = \varnothing.$$

A pixel present in only one foreground contributes zero. Depth agreement is the mean over four views. The combined score is

$$G = 0.40\,\mathrm{IoU}_{\mathrm{voxel}} + 0.35\,\mathrm{IoU}_{\mathrm{silhouette}} + 0.25\,D.$$

These references are the paired brick programs, not source meshes or RGB images.

B.3 Structural terms and solver mapping

Two bricks have a vertical contact edge if their layers differ by one and their XY footprints overlap with positive area. Ground reachability C is the fraction of bricks reachable in this undirected graph from any brick at $z = 0$. All bricks can be ground-reachable without forming one connected component. The support proxy I is the fraction of non-ground bricks touching at least two distinct lower-layer bricks; it is zero when there are no non-ground bricks. The offset proxy J is the fraction of vertical-contact pairs whose (h, w, x, y) tuples differ; it is zero when there are no such pairs.

The static worker constructs a `BrickStructure` and obtains a voxel score map. Let q be its maximum score over occupied cells. Its floating-brick check marks a non-ground brick unsupported if neither the layer above nor the layer below overlaps its footprint. With no worker exception, the

solver's binary label is $\neg\text{floating} \wedge q < 1$. The static reward component is one for a stable program and otherwise $S = \text{clip}(1 - q, 0, 1)$. On an exception it is zero. The structural score is

$$P = 0.30S + 0.25C + 0.25I + 0.20J.$$

The latter three terms are structural proxies, distinct from the binary static label.

The solver uses $g = 9.8$ and a connector threshold $T = 100$. The latter is converted internally to $Tg/1000 = 0.98$. Brick-unit dimensions are 0.0078 horizontally and 0.0096 vertically, and objective weights $\alpha = 10^{-3}$ and $\beta = 10^{-6}$. Brick masses are read from the retained brick library. A bottom cavity at $z = 0$ is treated as connected to ground; otherwise it connects to occupied cells immediately below. The solver models normal forces and connector forces using three or four connection points, depending on footprint width. Non-optimal solver status returns an all-one score map and therefore fails the stability threshold. The wrapper uses a 300-second request timeout.During online training, if the static-analysis worker times out or fails, we attempt to restart it and score the affected candidate using a fallback that sets the static-stability component S to zero. Geometric agreement G and the computable structural proxies C, I, and J are retained. Specifically, the physical reward becomes

$$P_{\text{fallback}} = 0.25C + 0.25I + 0.20J,$$

without renormalizing the remaining weights. The validity checks and geometry gate remain unchanged, and this fallback does not certify the candidate as physically stable. Worker-level fallback events are recorded explicitly.

The library masses in kilograms are 0.00043 (1×1), 0.00081 (1×2), 0.00115 (2×2), 0.00157 (1×4), 0.00216 (2×4), 0.00228 (1×6), 0.00323 (2×6), and 0.00303 (1×8); rotated footprints use the same mass.

B.4 Reward branches

As shown in algorithm 1, the exact reward used by the reported GRPO policy is

$$R = \begin{cases} -1 + 0.1G, & \text{invalid content,} \\ -0.5 + G, & \text{valid content and } G < 0.4, \\ 0.1 + 0.9(0.7G + 0.3P), & \text{valid content and } G \geq 0.4. \end{cases}$$

EOS is recorded during training but is not an additional argument in this reward function. A response ending on a complete line can therefore receive a content reward without EOS. The final evaluation protocol below separately requires sequence completion. The training gate 0.4 and evaluation Joint threshold 0.6 serve different purposes.

C Training and Implementation Details

C.1 SFT conditioning and prompts

The mixed-mode probabilities are 0.65 for RGB-D plus caption, 0.20 for RGB-D plus a generic instruction, and 0.15 for caption only. Caption-only inputs contain no images. In image-conditioned training, each view is independently dropped with probability 0.10: its RGB and encoded-depth images are both replaced by black images while retaining the view labels.

The multimodal system prompt is:

```
You convert multiview object observations into valid LEGO brick programs.
```

The caption-conditioned user text is:

```
Use the supplied front, right, back, and left views to infer the object's
↪  complete 3D shape. Object description: {caption}
 Each RGB view is followed by a depth encoding: red is normalized depth,
 ↪  green is the foreground mask, and blue is inverse normalized depth
 ↪  inside the mask.
Generate only the LEGO brick list. Use one brick per line in exactly this
↪  form: HxW (x,y,z). Coordinates must be integers in the 20x20x20
↪  construction grid. Do not add prose, numbering, or a code fence.
```

Algorithm 1 Reward Computation

Require: Generated brick program y; reference occupancy $O^\star$
Ensure: Scalar reward R
$(B, e_{\text{parse}}) \leftarrow \textsc{ParseAllNonblankLines}(y)$
if e_{parse} **then**
  $O \leftarrow \varnothing$
  $v \leftarrow \text{false}$
else
  $(O, e_{\text{bounds}}, e_{\text{overlap}}) \leftarrow \textsc{VoxelizeAndCheck}(B)$
  $v \leftarrow \textsc{ContentValid}(B, e_{\text{bounds}}, e_{\text{overlap}})$
end if
$G \leftarrow \textsc{GeometricAgreement}(O, O^\star)$
if $\neg v$ **then**
  return $-1 + 0.1G$
end if
$(C, I, J) \leftarrow \textsc{StructuralMetrics}(B)$
$S \leftarrow \textsc{StaticWorker}(B)$
$P \leftarrow \textsc{PhysicalScore}(C, I, J, S)$
if $G < 0.4$ **then**
  return $-0.5 + G$
else
  return $0.1 + 0.9(0.7G + 0.3P)$
end if

For generic conditioning, the first sentence is replaced with "Use the supplied canonical views to infer the object's complete 3D shape." For caption-only conditioning, the system refers to "object descriptions" instead of "multiview object observations." The native Qwen processor and chat template insert image tokens and assistant termination tokens. Cross-entropy labels mask the prompt and padding; only the assistant response is supervised. The 8,192-token SFT limit covers prompt plus response, with right truncation. A prompt that already reaches this limit is rejected.

Table 6: Settings of the reported policies. LoRA targets are `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`, and `qkv`.

Setting	Multimodal SFT	Online GRPO
Base model	Qwen3-VL-2B-Instruct	Same base and SFT adapter
LoRA	Rank 32, alpha 64, dropout 0.05	Initialized from SFT; reference frozen
Adaptation	Language projections and vision QKV	Language adapter; visual parameters frozen
Learning rate	2×10^{-4}	5×10^{-7}
Schedule	Cosine, 3% warmup	Cosine, 100-step warmup, horizon 10,651
Batch	1/rank, 8 ranks, accumulation 1	1 prompt/rank, 4 ranks, 4 candidates/prompt
Update count	One update/batch	Two optimizer updates/rollout step
Precision	BF16; gradient checkpointing	BF16; gradient checkpointing; SDPA
Length limit	8,192 total tokens	3,072 new tokens/candidate
Generation	Teacher-forced targets	Temperature 0.8; top-p 0.95
Regularization	Weight decay 0.01	KL 0.02; ratio clip 0.20; weight decay 0
Gradient norm	1.0	1.0
Seed	42	20260831
Checkpoint	Step 53,256, epoch 2	Step 10,000

SFT permits up to eight epochs, evaluates and saves after each epoch.

The SFT checkpoint used in our experiments was obtained using eight NVIDIA RTX 3090 GPUs, while online GRPO training used four NVIDIA RTX 4090 GPUs, each with 24 GB of memory. The collected environment snapshot contains PyTorch 2.7.1, PEFT 0.17.1, and the Transformers 4.57.6 overlay for the training process; the separate physics environment contains Gurobi 12.0.2. The package snapshot and source hashes are retained with the implementation material.

C.2 Online updates and short-horizon ablations

Each rank samples four candidates from its current policy for one condition, scores them, and computes group advantages using the population standard deviation clamped below at 10^{-4}. The behavior log probabilities remain fixed for the two optimization passes on that rollout. The token ratio is $\exp(\mathrm{clip}(\log \pi_\theta - \log \pi_{\mathrm{old}}, -20, 20))$. The KL estimate is $\exp(\Delta) - \Delta - 1$ with $\Delta = \mathrm{clip}(\log \pi_{\mathrm{ref}} - \log \pi_\theta, -20, 20)$. The objective averages over response tokens and candidates; distributed gradients are synchronized across four ranks. New candidates are drawn after the two updates. The reference remains the second-epoch SFT policy. The manifest is shuffled using the recorded seed without category balancing or unique-source-object sampling.

One rollout step processes four conditions and sixteen candidates across the four GPUs. Thus 500/1,000 steps correspond to 2,000/4,000 condition presentations, 8,000/16,000 candidates, and 1,000/2,000 optimizer updates. Short-run variants retain the original 10,651-step learning-rate horizon rather than compressing the schedule. The control is the corresponding early checkpoint from the main run. The geometry-only reward is $0.1 + 0.9G$ for valid content; both retain the same invalid-content branch.

D Evaluation Protocols and Baseline Adaptations

D.1 Direct generation and metric denominators

SFT and GRPO use one greedy response per condition, up to 8,192 new tokens, with no grammar, collision filter, or physical correction. The evaluator records EOS, full-text format, content validity, and solver results separately. Reported direct Valid requires valid content and EOS. Stable requires valid content, EOS, and successful static verification. Joint@0.6 additionally requires $G \geq 0.6$. Visual and voxel IoU are averaged over all 250 objects; non-EOS and audit-error outputs contribute zero. Parse errors already have empty scored occupancy. Parseable colliding or out-of-bounds outputs retain their computed geometry scores when completed and successfully audited. No stable prefix replaces the raw response in the direct-generation evaluation.

D.2 Budgeted physical search

As shown in algorithm 2, the search decoder restricts token choices with the brick-line grammar. After each complete line, it checks vocabulary, bounds, collision, and repeated rejected placements. A rejected line restores the cache to its preceding boundary. Temperature starts at 0.6 and increases by 0.01 on a repeated rejected placement, up to 2.0. At EOS, the full accepted program is checked by the static worker. If unstable, the worker identifies the first brick whose occupied cells have score at least one and removes that brick and the suffix. It repeats assessment until reaching a stable prefix; if no offending brick is found it removes the last brick. Generation resumes from the retained prefix.

The caps are 32,000 sampled tokens, 24,000 accepted-prefix tokens, 2,000 bricks, 100 suffix regenerations, and 500 rejected-line retries (the loop includes retry index zero). Rejected and subsequently discarded tokens count toward the sampled budget, including sampled EOS tokens. No target program or target geometry score is consulted by this search. GT-dependent geometry is computed afterward.

The 4K/8K/16K/32K budget points are obtained from EOS snapshots of each saved 32K trajectory, not independent re-runs. If no completed snapshot exists within a budget, all primary success and geometry fields are zero. An unstable completed snapshot can retain a nonzero geometry score. Ground reachability was additionally logged; all outputs counted as stable in the audited results passed this check. Mean cost includes all 250 objects and uses $\min(\text{sampled counter}, B)$. Search time includes prefill, generation, and online verification, but excludes model loading and the post-search GT audit. Lower-budget times are not interpolated from final-trajectory times.

D.3 Baselines and input transformations

Hunyuan3D-2mv + Mesh2Brick. We compose the multi-view shape model from Hunyuan3D 2.0 (Zhao et al., 2025) with the Mesh2Brick conversion module released in the BrickGPT repos-

Algorithm 2 Budgeted Generation with Physical Rollback

Require: Policy π_θ; observations $\mathcal{O}$; text c; search caps; reporting budget B
Ensure: A complete program within budget B, or $\varnothing$

Initialize empty prefix p, KV cache $\mathcal{K}$, and snapshot list $\mathcal{S}$
Initialize sampled-token counter $n \leftarrow 0$ and rejection/rollback counters
while no sampled-token, accepted-token, brick, rejection, or rollback cap is reached **do**
    $t \leftarrow \textsc{SampleGrammarPermittedToken}(\pi_\theta, \mathcal{O}, c, p, \mathcal{K})$
    $n \leftarrow n + 1$ ▷ Includes rejected/discarded tokens and EOS
    Append t to p and update $\mathcal{K}$
    if a brick boundary is reached and the placement overlaps or is out of bounds **then**
        Reject the placement, restore p and $\mathcal{K}$, and increment the rejection counter
    else if $t = \text{EOS}$ **then**
        Save an immutable snapshot $s = (p, n, \text{unverified})$ in $\mathcal{S}$
        $r \leftarrow \textsc{StaticVerify}(p)$
        if $r =$ stable **then**
            Mark s as verified stable; **break**
        else if $r =$ worker error **then**
            Record the error; **break**
        else
            Mark s as unstable
            Retain the stable prefix of p and restore its KV cache $\mathcal{K}$
            Increment the rollback counter and resume suffix generation
        end if
    end if
end while
$\mathcal{S}_B \leftarrow \{s \in \mathcal{S} : s.\text{tokens} \leq B\}$ ▷ Filter the saved trajectory
if $\mathcal{S}_B$ contains a verified stable snapshot **then**
    return the program of the first verified stable snapshot in $\mathcal{S}_B$
else if $\mathcal{S}_B \neq \varnothing$ **then**
    return the program of the last complete snapshot in $\mathcal{S}_B$
else
    return $\varnothing$
end if

Table 7: Inputs and generation-time constraints of the evaluated implementations.

Method	Input and conversion	Generation constraints/assistance
BrickGPT	Fixed caption; original model	Brick grammar and local placement rejection; optional static-prefix rollback
STABLE	Point cloud sampled from the ground-truth mesh and serialized in STABLE's native input format	Greedy autoregressive decoding; no physical rollback
TreeSBA	Silhouettes of observed RGB-D-derived occupancy in its native centered grid	Native action checks plus workspace rejection; no static solver during generation
Hunyuan3D-2mv + Mesh2Brick	Multi-view RGB to mesh, then mesh-to-brick conversion	Physically assisted conversion pipeline
VISTA-Brick	Four RGB-D views plus caption	Unconstrained greedy direct decoding, or the specified physical search

itory (Pun et al., 2025). The Hunyuan3D citation applies to mesh reconstruction; the subsequent physics-assisted brick conversion is a separate stage of our integrated baseline.

Mesh reconstruction uses four RGB views with the recorded foreground masks as alpha channels, without depth, captions, or target geometry. We run the native flow-matching sampler in FP16 with 50 inference steps, octree resolution 380, `num_chunks=20000`, and seed $20260912 + i$ for zero-based manifest index i. Texture generation is not used. The frozen conversion backend is copied from the `src/mesh2brick` module of `https://github.com/AvaLovelace1/BrickGPT`. For each reconstructed mesh, we run one conversion attempt in a $20 \times 20 \times 20$ workspace with `max_failures=10`, retaining the backend's stability optimization. Our coor-

dinate adapter applies a 90° X rotation before voxelization and the fixed upper-Y anchoring convention, using generated voxel bounds rather than a GT-fitted alignment. All 250 objects remain in the evaluation denominator; failed conversions are not replaced by successful prefixes. This is a physically assisted conversion baseline, not an autoregressive token-budget-matched method.

STABLE. We sample a point cloud from the ground-truth mesh and serialize it in STABLE's native input format. This baseline therefore receives oracle geometry through the ground-truth mesh, rather than a point cloud backprojected from the four RGB-D views. Generation uses greedy autoregressive decoding without physical rollback.

TreeSBA uses its frozen ModelNet40 (Sun et al., 2022) checkpoint and native $2\times4/4\times2$ vocabulary with a 201-brick cap. The observed median root is clamped to a feasible root location; the resulting translation is reversed when mapping output bricks back to the benchmark frame. Workspace checks reject out-of-bounds actions.

E Additional Experiments and Diagnostics

E.1 Budget curves and search-cost decomposition

Table 8: Budget sensitivity reconstructed from saved search trajectories. Stable and Joint are percentages, always using 250 objects.

Policy	Budget	Stable	Visual	IoU	Joint
SFT	4K	66.0	0.6771	0.5706	61.2
GRPO	4K	**76.0**	**0.7269**	**0.6097**	**71.2**
SFT	8K	74.0	**0.7598**	0.6276	69.2
GRPO	8K	**83.2**	0.7563	**0.6286**	**77.6**
SFT	16K	80.0	0.7618	0.6290	75.2
GRPO	16K	**89.2**	**0.7640**	**0.6348**	**82.8**
SFT	32K	84.4	0.7648	0.6317	79.2
GRPO	32K	**94.0**	**0.7648**	**0.6367**	**87.2**

At 32K, stability increases from 84.4% to 94.0% and Joint@0.6 from 79.2% to 87.2%. The budget-accounted token means are 7,555.14 and 5,015.872. Table 9 breaks down the recorded search process. GRPO-10K has fewer suffix regenerations and verifier requests, and its returned completed structures are also shorter. Verification time does not decrease with request count: the recorded mean is 25.45 versus 33.51 seconds. The time measurements are from the corresponding runs, whereas sampled tokens are the defined budget unit.

Table 9: Search diagnostics averaged over all 250 objects, including failures. The terminal internal prefix can differ from the completed snapshot returned for evaluation. Raw counters can slightly exceed the budget-accounted counter.

Quantity (mean over 250)	SFT	GRPO-10K
Raw sampled token counter	7555.14	5015.88
Verifier requests	5.34	3.41
Suffix regenerations	4.47	2.47
Rejected brick lines	69.48	62.24
Stability API calls	17.62	11.01
Score API calls	6.98	4.19
Returned complete bricks	145.32	125.04
Terminal internal-prefix bricks	131.06	122.56
Search seconds	396.94	274.72
Verification seconds	25.45	33.51

F Qualitative Outputs and Scope

The accompanying examples use the same SFT and GRPO checkpoints as the main tables. The first object is fixed at manifest index 0. Each subsequent case is the lowest unused manifest index satisfying a stated output predicate: SFT unstable/GRPO stable (1), SFT stable/GRPO unstable (18), GRPO valid with $G \geq 0.6$ but unstable (20), GRPO stable with $G < 0.6$ (4), and GRPO search not stable by 32K (10). These are diagnostic cases, not a random visual sample. Each stored case contains its caption, full GT program, four generated programs, and per-program scores. No output from a different GRPO checkpoint is substituted.

The reported setting uses calibrated synthetic observations, a fixed 20^3 workspace, and the specified brick vocabulary. Stable is the result of the adopted static model, and geometric agreement is measured against a discrete reference assembly.

Table 10: Per-category manifest counts. Source-test and evaluation counts are unique objects. Category 02843684 is absent only from the selected evaluation set.

Category ID	Train objects	Train structures	Eval objects
02801938	89	159	1
02818832	193	290	15
02828884	1550	2514	15
02843684	61	109	0
02871439	362	617	15
02876657	437	854	15
02880940	136	248	6
02924116	830	1396	14
02942699	88	159	1
02958343	3100	5896	15
03001627	5294	8634	15
03467517	682	1361	15
03593526	389	732	15
03797390	159	278	14
03928116	196	304	13
03991062	293	525	15
04256520	2825	4955	15
04379243	7089	9986	15
04460130	100	190	8
04468005	340	658	14
04530566	1440	2739	14

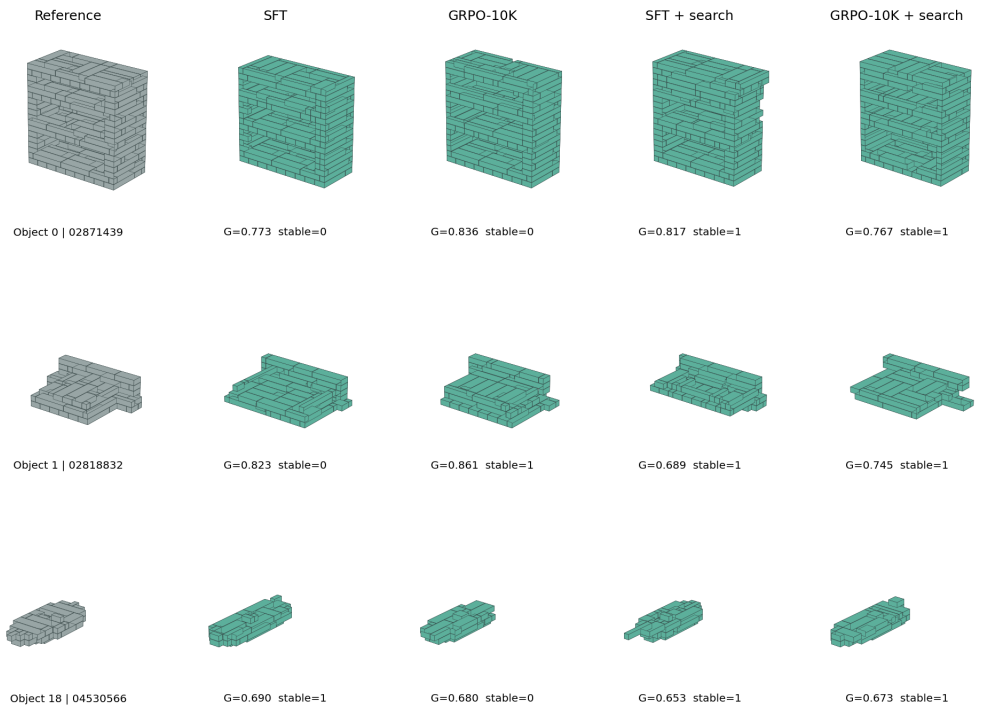

Figure 7: Fixed diagnostic objects 0, 1, and 18. Uniform cuboid renders show the stored brick volumes in the same workspace and camera; gaps expose individual brick boundaries. Search columns use the returned 32K EOS snapshot.

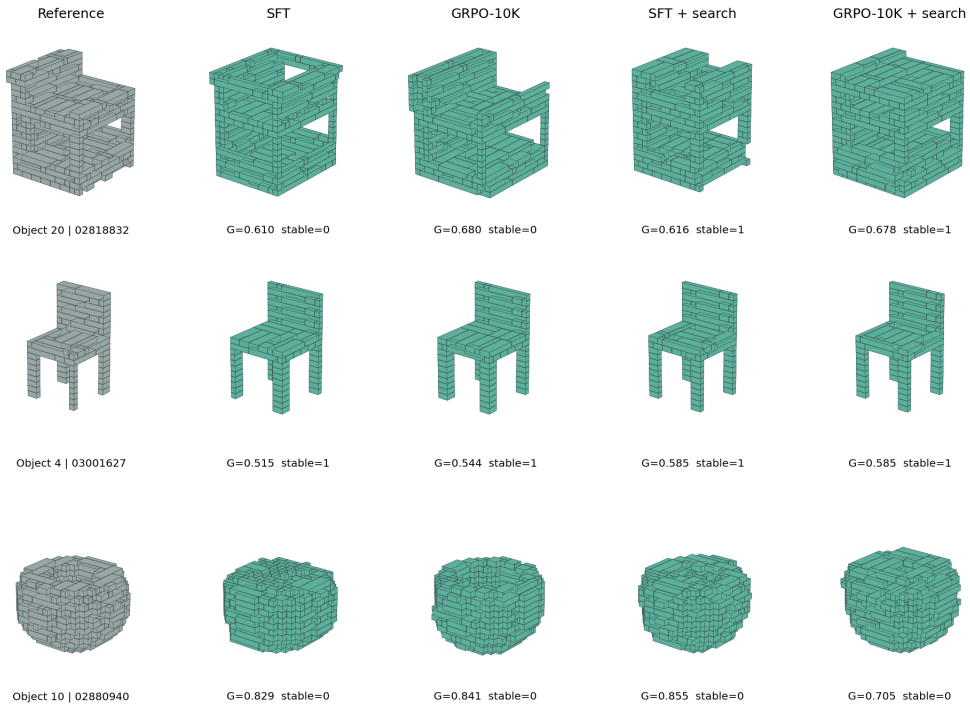

Figure 8: Diagnostic objects 20, 4, and 10, covering high geometric agreement without static stability, stability below the geometric threshold, and search failure within the budget. A visible discrepancy alone is not assigned a causal label.