# BRICK-COMPOSER: MLLMS CONSTRUCT EVERYTHING FROM BUILDING BLOCKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We dream of AI agents that can read arbitrary designs and construct real-world objects from reusable building blocks. As a first step toward this vision, we study whether multimodal large language models (MLLMs) possess the visual grounding and spatial reasoning capabilities required for brick assembly. We formulate brick assembly as a sequential decision-making problem, where each step involves two subtasks: *brick selection*, identifying the target brick from candidate components, and *brick pose estimation*, predicting where and how the selected brick should be placed. To support this study, we introduce **BC-Bench** (**B**rick **C**onstruction **Bench**mark), the first benchmark for evaluating MLLMs on assembly with diverse bricks. Experiments show that current state-of-the-art MLLMs remain far from reliable builders, struggling with fine-grained brick selection and failing at precise pose estimation. To bridge this gap, we propose **Brick-Composer**, a learning framework that equips MLLMs with assembly skills through three complementary signals: *Human Design Sparks*, which provide affordance-rich construction demonstrations; *World Feedback*, which grounds predicted actions in visual and physical consequences; and *Synthetic Experience*, which scales learning beyond existing object designs. Brick-Composer improves brick selection accuracy by over three times, substantially reduces pose estimation errors, and raises strict step-level assembly success from less than 1% to around 15%. After training, a Qwen-3-VL-8B can correctly compose up to 42% of the steps for a complete object, suggesting that MLLMs can acquire assembly capabilities through targeted, physically grounded learning.

## 1 INTRODUCTION

It is such a beautiful dream to imagine a future where robots can read arbitrary designs and simply construct almost anything from reusable building blocks: a piece of furniture, a car, a house, or even an aircraft. Yet realizing this dream, and achieving such a level of generalization, requires a surprisingly deep form of intelligence that goes far beyond visual understanding alone, and we are still only at the very early stages of reaching it Pun et al. (2025); Tie et al. (2025b); Kulits & Schmid (2026); Jing et al. (2026). A capable assembly agent must recognize building blocks from fine-grained geometric features and functional affordances, infer each brick's target pose within the correct assembly sequence, execute reliable actions in the physical world, and continuously monitor the long-horizon construction process while remaining able to recover from errors along the way.

In this paper, we use LEGO-style assembly as a proxy task for pursuing this long-term vision, taking an initial step toward enabling multimodal large language models (MLLMs) to reason about complex object construction. As illustrated in Figure 1, we formulate assembly with diverse bricks as a sequence of step-wise decisions. At each step, the model observes the instructional manual and is required to solve two coupled subtasks: brick selection, which identifies the next brick from a set of candidates, and brick pose estimation, which predicts where and how the selected brick should be placed based on the current assembly state and the brick's geometry. This setting closely reflects how humans follow assembly manuals: interpreting visual instructions, choosing the appropriate component, and integrating it into an evolving structure with the correct spatial pose.

To support this study, we introduce **BC-Bench** (**B**rick **C**onstruction **Bench**mark), the first benchmark for evaluating MLLMs on assembly with diverse bricks. Built from human-designed LEGO-style



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.

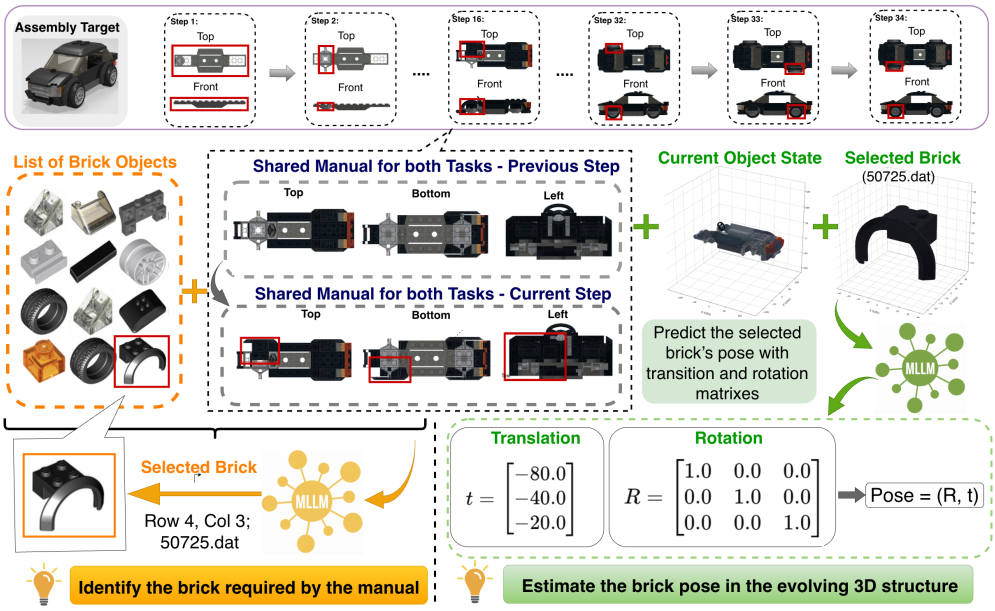


Figure 1: Overview of the BC-Bench task setting. **Left: Brick selection.** The model selects the required brick from a candidate grid using manual images. **Right: Brick pose estimation.** Given the manual context, current assembly state, and selected brick, the model predicts the brick's target pose as a translation vector and rotation matrix.

objects and diverse brick inventories, BC-Bench uses a custom simulator to reconstruct step-wise assembly processes and render multimodal observations, including manual-style instruction views, candidate brick views, and evolving assembly states from orthogonal and isometric viewpoints. This design enables controlled evaluation of brick selection and pose estimation while preserving the structural complexity, design diversity, and real-world affordances of human-created assemblies. Using BC-Bench, we conduct a comprehensive evaluation of whether current MLLMs can perform LEGO-style assembly with diverse bricks, covering both fine-grained brick selection and brick pose estimation. The results reveal a clear gap between current model capabilities and the demands of reliable assembly: even state-of-the-art MLLMs struggle to identify the correct brick at each step, and most models produce large translation and rotation errors when asked to place the selected brick.

Despite these limitations, we introduce **BrickComposer**, a learning framework that equips MLLMs with assembly skills through three complementary forms of supervision: *Human Design Sparks*, *World Feedback*, and *Synthetic Experience*. Together, these signals capture three key merits of assembly learning: design-driven structures grounded in real-world affordances, simulator-based feedback for error discovery and recovery, and scalable compositional diversity beyond existing object designs. *Human Design Sparks* provide high-quality demonstrations from human-designed objects, showing how diverse bricks are used to form meaningful structures and support different object functions. While these examples teach the basic language of assembly, they are valuable but scarce compared to the large sum of structures we could build with 9,876 different bricks. From all the 9,977 structures we obtained from the benchmark, we only select 82 human-designed objects, yielding 3,873 step-wise training examples as gold demonstrations. *World Feedback* grounds model predictions in their visual and physical consequences by executing predicted selections or poses in our simulator, rendering the resulting assembly state, and highlighting the predicted brick or placement. This exposes model errors and supports recovery, either through inference-time revision or training-time correction. *Synthetic Experience* scales learning beyond existing designs by generating physically plausible brick configurations through feasible attachments, while filtering for collision-free and compact structures. We produce synthetic objects with 20 to 100 pieces and yields around 40K step-wise training examples, exposing models to broader geometries, connection affordances, and attachment patterns.

Together, these components provide a progressive path for assembly learning: from interpreting human designs, to correcting simulated actions, and generalizing across scalable physically grounded

Table 1: Comparison of BC-Bench with related assembly and LEGO-style benchmarks. ✓indicates the feature is explicitly supported; ✗indicates it is not the focus; △indicates partial or related support.

| Benchmark / Dataset | Domain | Diverse Parts | Manual-based | Pose Est. w/ Rotation |
|---|---|---|---|---|
| LEGO Manual (Wang et al., 2022) | LEGO Manual Understanding | △ | ✓ | ✗ |
| LEGO-Puzzles (Tang et al., 2025) | LEGO Spatial Reasoning | ✗ | △ | ✗ |
| BrickGPT (Pun et al., 2025) | Text-to-brick Generation | ✗ | ✗ | ✗ |
| LEGO Co-builder (Huang et al., 2025) | LEGO assembly Assistant | △ | ✓ | ✗ |
| BrickNet (Kulits & Schmid, 2026) | Brick Sequence Generation | ✓ | ✗ | ✗ |
| AssemLM (Jing et al., 2026) | Furniture Assembly | △ | ✓ | △ |
| Manual2Skill (Tie et al., 2025b) | Furniture Assembly | ✗ | ✓ | △ |
| **BC-Bench (Ours)** | Diverse LEGO Brick Assembly | ✓ | ✓ | ✓ |

configurations. Empirically, Brick-Composer turns LEGO-style assembly from an almost impossible zero-shot challenge into an emerging capability. Step-wise brick selection accuracy increases from roughly 23% with base Qwen-3 models to around 70% after learning, while average translation and rotation errors are significantly reduced . Under a strict criterion requiring both the correct brick and an accurate pose, base MLLMs achieve nearly 0% successful assembly steps, whereas our learned models can reach an average of 15% and peaks at 42%. This indicates that MLLMs can already complete a non-trivial subset of construction steps by selecting the right component and placing it into an acceptable target pose. Although fully autonomous assembly remains far from solved, these results suggest that assembly is a learnable capability: with larger-scale human supervision, stronger visual feedback, and higher-quality physically grounded data, MLLMs may gradually move from reading assembly instructions to executing them reliably.

## 2 Benchmarking MLLMs for Assembly

We consider a step-wise brick assembly scenario in which an MLLM serves as an assembly agent in a simulated construction environment. Given a complex target object, the model observes the corresponding assembly manual, the current partially assembled structure, and a workspace containing candidate bricks with diverse geometries, functions, and physical affordances. As shown in Figure 1, each assembly step requires the model to solve two coupled problems. First, in ***Brick Selection***, the model identifies the brick specified by the current manual instruction from the available candidate set. Second, in ***Brick Pose Estimation***, the model predicts the selected brick's 3D placement, including its orientation and position relative to the existing structure. The process repeats iteratively until the target object is completed. We compare our benchmark with existing assembly, LEGO, and spatial reasoning benchmarks in Table 1. Appendix A details collection, the object-level split, and held-out-design coverage. Figures 5 and 6 illustrate assembly sequences and local-frame part renderings. Catalog sizes, shuffling, and interchangeable target instances are described there, with inventory and repetition statistics in Table 5.

### 2.1 Brick Selection

As shown in Figure 1 (Left), the brick selection task evaluates whether an MLLM can understand the visual assembly manual and identify the required brick from its geometric features and functional affordances. At each step, the model is given the assembly manual, including current-step and target-step views rendered from six orthogonal perspectives, together with a visual grid of candidate bricks. We formulate brick selection as grounded localization over this grid, where the model predicts the row and column of the target brick. Formally, at assembly step $t$, let $M_t$ denote the manual context, including the current-step view and the target-step view. Let the candidate brick grid be denoted as

$$\mathcal{B}_t = \{b_t^{i,j} \mid 1 \le i \le H, 1 \le j \le W\}, \tag{1}$$

where $b_t^{i,j}$ represents the candidate brick located at row $i$ and column $j$ in a grid with $H$ rows and $W$ columns. The ground-truth target brick is represented by its grid coordinate: $y_t^* = (i_t^*, j_t^*)$. Given the manual context $M_t$, the candidate grid $\mathcal{B}_t$, and a task prompt $P_1$, the MLLM parameterized by $\theta$ predicts the grid coordinate of the brick required for the current step:

$$\hat{y}_t = (\hat{i}_t, \hat{j}_t) = \mathrm{MLLM}_\theta(P_1, M_t, \mathcal{B}_t). \tag{2}$$

The selected brick is therefore given by $\hat{b}_t = b_t^{\hat{i}_t, \hat{j}_t}$. This task captures a common requirement in assembly. The challenge of the task increases when the workspace contains many visually similar

parts that share colors or shapes but differ in subtle geometry or affordances, which are analyzed in Section 4.

### 2.2 BRICK POSE ESTIMATION

As shown in Figure 1 (Right), the brick pose estimation task evaluates whether an MLLM can infer how the selected brick should be placed in the current state under the correct environmental scale. At each step, the model is given the manual, the partially assembled structure, and the geometric details of the selected brick. The model must predict the brick's translation and rotation in the target-object coordinate system. Formally, at assembly step $t$, let $M_t$ denote the manual context, including the current-step view and the target-step view. Let $S_t$ denote the current partially assembled structure in the simulated environment, represented by multi-view renderings under the target-object coordinate system $\mathcal{C}$. As shown in Figure 1, this includes six orthogonal views and one isometric view: $S_t = \{v_t^1, v_t^2, \ldots, v_t^K\}$, where each $v_t^k$ is a 2D rendering from a predefined camera view, and $K = 7$ in our setting. This representation exposes the scale and spatial layout of the current structure. Similarly, let $b_t$ denote the selected brick, whose geometry is rendered in its intrinsic coordinate system $\mathcal{C}_{b_t}$ using the same multi-view format: $b_t = \{g_t^1, g_t^2, \ldots, g_t^K\}$, where each $g_t^k$ shows the selected brick from the corresponding view, revealing its geometry, connection surfaces, and orientation-sensitive affordances. The goal is to predict the absolute pose of the selected brick in $\mathcal{C}$. We represent the ground-truth pose as a rigid transformation:

$$P_t^* = \begin{bmatrix} R_t^* & T_t^* \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3), \tag{3}$$

where $R_t^* \in SO(3)$ is the ground-truth rotation matrix and $T_t^* \in \mathbb{R}^3$ is the ground-truth translation vector. The translation vector and the rotation matrix are defined as:

$$T_t^* = \begin{bmatrix} x_t^* \\ y_t^* \\ z_t^* \end{bmatrix}, \quad R_t^* = \begin{bmatrix} r_{11}^* & r_{12}^* & r_{13}^* \\ r_{21}^* & r_{22}^* & r_{23}^* \\ r_{31}^* & r_{32}^* & r_{33}^* \end{bmatrix}, \tag{4}$$

where $x_t^*$, $y_t^*$, and $z_t^*$ specify the target position, and $\{r_{mn}^*\}_{m,n=1}^3$ define the rotation from the brick's intrinsic coordinate system $\mathcal{C}_{b_t}$ to the target-object coordinate system $\mathcal{C}$. Given the task prompt $P_2$, the manual context $M_t$, the current assembly observation $S_t$, and the selected-brick geometry observation $b_t$, the MLLM parameterized by $\theta$ predicts the absolute target pose:

$$\hat{P}_t = \begin{bmatrix} \hat{R}_t & \hat{T}_t \\ \mathbf{0}^\top & 1 \end{bmatrix} = \mathrm{MLLM}_\theta(P_2, M_t, S_t, b_t). \tag{5}$$

This task captures a realistic spatial reasoning requirement in assembly: predicting where and how a selected component should be placed within the target-object coordinate system. The formulation provides a unified, model-agnostic interface that is interpretable, directly evaluable in simulation, and compatible with downstream robotic execution.

## 3 METHODS

**Direct Prompting.** We first evaluate whether general-purpose MLLMs can perform assembly through multimodal prompting. At each step, the model receives visual inputs for brick selection or pose estimation, together with task instructions and reasoning guidance. Appendix B provides brick-selection, pose-estimation, and world-feedback trajectories, including the input conventions and structured output formats. As shown in Table 2 and Section 4.3, prompting alone remains insufficient for reliable brick selection and precise pose prediction, motivating our learning framework with *Human Design Sparks*, *World Feedback*, and *Synthetic Experiences* shown in Figure 2.

### 3.1 LEARNING FROM HUMAN DESIGN SPARKS

To move beyond zero-shot prompting, we adapt MLLMs with supervised data from human-designed brick assemblies. We call these data *Human Design Sparks* because they capture not only valid placements, but also physically meaningful structures, feasible connections, creative construction

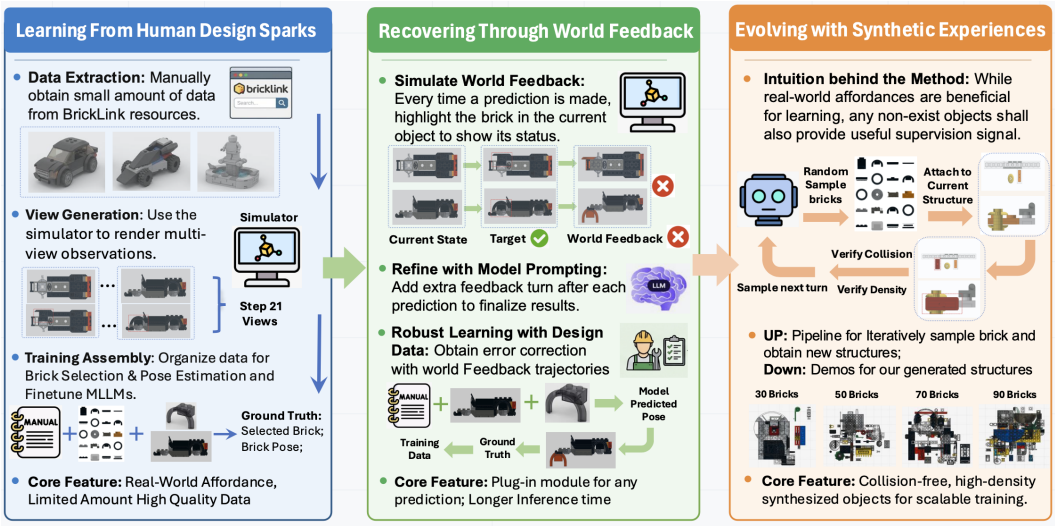


Figure 2: Overview of the **Brick-Composer** learning framework. We improve assembly reasoning through three complementary signals: Human design supervision, World feedback for error recovery, and Scalable synthetic objects for experience expansion. Together, they substantially enhance the model's assembly capabilities.

patterns, and final objects with real-world affordances. Such signals are difficult to obtain from random configurations alone and help the model connect its real-world priors with concrete assembly decisions. We use our simulator to convert symbolic annotations into MLLM-ready visual inputs: six-view orthogonal before-and-after manuals with the current brick highlighted, and coordinate-augmented renderings of the current state and selected brick for spatial prediction.

We train both brick selection and pose estimation by serializing task outputs as natural-language answers and applying a standard autoregressive next-token objective. Each assembly step is treated as an independent training instance. For step $t$, let $V_t$ be the visual input, $I$ the task instruction, and $A_t = \{a_1, \ldots, a_K\}$ the answer token sequence. The objective is

$$\mathcal{L}_{\text{Designer}} = -\sum_{k=1}^{K} \log p_\theta \left(a_k \mid a_{<k}, I, V_t\right). \tag{6}$$

The target answer is $A_t^{\text{sel}} = y^* = (i^*, j^*)$ for brick selection and $A_t^{\text{pose}} = P^* = (R^*, T^*)$ for pose estimation, where $R^* \in SO(3)$ and $T^* \in \mathbb{R}^3$. Thus, both tasks are optimized under the same autoregressive prediction framework.

### 3.2 Learning from World Feedback

Human assembly is rarely completed through a single prediction; constructors place a part, inspect the resulting structure, and correct mistakes. We introduce a similar world-feedback mechanism for MLLMs. As shown in Figure 2, our simulator approximates a physical assembly environment by executing the model's predicted selection or pose, rendering the resulting state, and highlighting the selected brick and placement. The model can therefore observe the consequence of its action and use visual discrepancies as a feedback.

Formally, at step $t$, the model receives visual input $V_t$ and instruction $I_t$, generates an initial response $\hat{A}_t$, and obtains a simulator-rendered feedback observation $O_t$. We use feedback in two settings. For prompting, we append $O_t$ to the trajectory and ask the model to revise its prediction:

$$\hat{A}_t^{\text{revise}} = \text{MLLM}_\theta \left(I_t, V_t, \hat{A}_t, O_t\right). \tag{7}$$

For training, we construct correction trajectories from model errors and fine-tune the model to predict the corrected answer:

$$\mathcal{L}_{\text{Fb}} = -\sum_{k=1}^{K} \log p_\theta \left(a_k \mid a_{<k}, I_t, V_t, \hat{A}_t, O_t\right) \tag{8}$$

Table 2: Main evaluation of state-of-the-art MLLMs on BC-Bench. We report overall average and best-object performance, with higher-is-better metrics marked by ↑ and lower-is-better metrics marked by ↓. The results show that current MLLMs remain limited in reliable brick selection and precise pose estimation.

| Eval Metrics → | Overall Average | | | | Best Object Performance | | | |
|---|---|---|---|---|---|---|---|---|
| Model ↓ | Selection ↑ Acc. (%) | PE Trans. ↓ Err(LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) | Selection ↑ Acc. (%) | PE Trans. ↓ Err(LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) |
| Gemma-3-12B | 4.35 | 269.09 | 63.00 | 0.09 | 17.24 | 41.86 | **12.86** | 2.22 |
| InternVL-3.5-8B | 13.24 | 221.69 | 88.43 | 0.00 | 31.25 | **41.54** | 35.00 | 0.00 |
| Qwen-3-VL-8B | 22.76 | **210.14** | 62.47 | **0.36** | 55.17 | 43.81 | **12.86** | **4.44** |
| Qwen-3.5-VL-27B | 37.44 | 314.32 | 82.94 | 0.18 | 75.86 | 64.47 | 34.84 | **4.44** |
| GPT-5.4 | **43.88** | 310.78 | 74.85 | 0.18 | **93.10** | 67.94 | 40.00 | 2.22 |

This encourages the model to further learn to interpret world-state discrepancies and recover from wrong selections or inaccurate poses.

### 3.3 Scaling with Synthetic Experiences

Although human designs provide high-quality supervision, their scale is limited by the availability of human-designed models, object-specific semantics, and copyright constraints, making brick selection and especially pose estimation difficult to learn at scale. Since both tasks mainly rely on fine-grained visual grounding, part geometry, and spatial reasoning, we further scale learning with *Synthetic Experiences*: arbitrary but physically plausible configurations that expose the model to diverse brick placements and pose patterns beyond existing object designs. While these configurations lack real-world semantics and object-level affordances, they serve as a scalable source of assembly experience, enabling the model to practice a much broader range of local connections and spatial transformations.

We generate each configuration by sampling a new brick, attaching it to an existing partial structure through feasible connection points, and filtering placements by collision avoidance and structural connectivity. A density-based reward further encourages compact structures rather than sparse random layouts. Each accepted placement is treated as a synthetic assembly step, rendered in the same before-and-after visual format as designer supervision, and used for both brick selection and pose estimation. We scale this process to generate about 700 synthetic objects, each containing 20–90 assembly steps, yielding over 40K training steps and substantially expanding the supervision available for learning. Figure 7 illustrates the generated structures; Appendix C discusses their feasible connections and compactness, and Appendix F explains transfer from synthetic configurations to manual-conditioned assembly. We use the same training objective as designer supervision. For each synthetic step $t$, the model receives the rendered visual input $V_t^{\text{arb}}$, task instruction $I_t$, and target answer $A_t^{\text{arb}}$:

$$\mathcal{L}_{\text{Arb}} = -\sum_{k=1}^{K} \log p_\theta \left( a_k \mid a_{<k}, I_t, V_t^{\text{arb}} \right). \tag{9}$$

## 4 Experiments

### 4.1 Experiment Settings

We evaluate state-of-the-art MLLMs from different model families and scales on BC-Bench, including Gemma-3-12B (Team et al., 2025), InternVL3.5-8B (Wang et al., 2025), Qwen3-VL-8B (Bai et al., 2025), Qwen3.5-27B (Qwen Team, 2026), and GPT-5.4 (OpenAI, 2026). We further apply our learning framework to Gemma-3-12B and Qwen3-VL-8B as representative open-weight models. Appendix F specifies raw image resolutions, optimization, decoding, output parsing, and released datasets, code, splits, prompts, and seeds; Table 10 summarizes the training configuration. Appendix F also describes asset provenance and the additional components needed for physical transfer.

### 4.2 Evaluation Metrics

**Brick Selection Metrics** We report ***Set-based Accuracy*** to handle physically identical and interchangeable candidate bricks. At time step $t$, the model predicts $(\hat{b}_t^i, \hat{b}_t^j)$, which is counted as correct

Table 3: Performance improvements on BC-Bench. Our learning framework consistently improves brick selection accuracy, reduces pose estimation errors, and increases step-level assembly success. World Feedback (P) denotes direct prompting with simulator feedback, while World Feedback (L) denotes learning from simulator-generated feedback trajectories jointly with human supervision data. Brick-Composer integrates all the learning signals.

| Eval Metrics → | Overall Average | | | | Best Object Performance | | | |
|---|---|---|---|---|---|---|---|---|
| Approaches ↓ | Selection ↑ Acc. (%) | PE Trans. ↓ Err(LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) | Selection ↑ Acc. (%) | PE Trans. ↓ Err(LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) |
| **Learning Approaches on Gemma-3-12B** | | | | | | | | |
| **Direct Prompting** | 4.35 | 269.09 | 63.00 | 0.09 | 17.24 | 41.86 | **12.86** | 2.22 |
| **World Feedback (P)** | – | 273.41 | 62.26 | 0.00 | – | 39.46 | 14.53 | 2.22 |
| **Designer Supervision** | 15.59 | 201.51 | 55.69 | 0.72 | 26.67 | 36.91 | 15.00 | 4.44 |
| **World Feedback (L)** | – | 170.33 | 51.49 | 1.99 | – | 27.56 | **12.86** | 9.07 |
| **Brick-Composer** | **17.95** | **123.69** | **45.43** | **4.35** | **52.87** | **24.63** | **12.86** | **18.75** |
| **Learning Approaches on Qwen-3-VL-8B** | | | | | | | | |
| **Direct Prompting** | 22.76 | 210.14 | 62.47 | 0.36 | 55.17 | 43.81 | 12.86 | 4.44 |
| **World Feedback (P)** | – | 226.33 | 65.66 | 0.27 | – | 42.36 | 12.86 | 4.44 |
| **Designer Supervision** | 48.29 | 162.82 | 57.81 | 5.40 | 73.24 | 27.13 | 12.95 | 8.92 |
| **World Feedback (L)** | – | 137.26 | 52.65 | 6.24 | – | 21.46 | 11.64 | 15.36 |
| **Brick-Composer** | **68.21** | **65.63** | **37.97** | **14.27** | **90.65** | **14.29** | **0.00** | **41.63** |

if it belongs to the valid equivalent set $\mathcal{B}_t^* = \{(b^i, b^j) \in \mathcal{C}_t \mid \phi(b^i, b^j) = \phi(b_t^i, b_t^j)\}$, where $\mathcal{C}_t$ is the candidate set and $\phi(\cdot)$ maps each brick to equivalence-defining attributes such as part ID, color, and geometry. The brick selection accuracy is then defined as $\text{Acc}_{\text{brick}} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}\big((\hat{b}_t^i, \hat{b}_t^j) \in \mathcal{B}_t^*\big)$, where $N$ is the number of evaluated steps.

**Brick Pose Estimation Metrics** We evaluate the predicted absolute pose of the selected brick in the shared target-object coordinate system. At step $t$, let the predicted pose be $\hat{P}_t = (\hat{R}_t, \hat{T}_t)$ and the ground-truth pose be $P_t = (R_t, T_t)$. We measure ***Mean Translation Error*** as $e^{\text{trans}} = \frac{1}{N}\sum_{t=1}^{N} \|\hat{T}_t - T_t\|_2$. We measure ***Mean Rotation Error*** by the angular distance between rotations, where $\theta_t = \cos^{-1}\left(\frac{\text{Tr}(R_t^\top \hat{R}_t)-1}{2}\right)$ and $e^{\text{rot}} = \frac{1}{N}\sum_{t=1}^{N} \frac{180}{\pi}\theta_t$. Since many bricks have rotational symmetries, we compute rotation error in a symmetry-aware manner. Let $\mathcal{S}_t$ be the set of valid symmetry rotations for the brick at step $t$. We define $\theta_t^{\text{sym}} = \min_{S\in\mathcal{S}_t} \cos^{-1}\left(\frac{\text{Tr}((R_t S)^\top \hat{R}_t)-1}{2}\right)$ and report the final mean rotation error as $e^{\text{rot}} = \frac{1}{N}\sum_{t=1}^{N} \frac{180}{\pi}\theta_t^{\text{sym}}$. Symmetry-equivalent orientations are not penalized; Appendix E describes how equivalent poses are obtained from part files and explains the physical scale and strict criterion. Translation uses LDraw Units: 1 LDU $= 0.4$ mm, and one stud pitch is 20 LDU.

**Joint Assembly Evaluation Metrics** We further evaluate joint performance using ***Averaged Step-level Success Rate***, where a step is successful only if the model selects the correct brick and predicts the target pose exactly within our simulated environment. Although practical systems may use collision handling to correct small pose deviations, we adopt a strict zero-tolerance criterion. We report all metrics from two perspectives: *overall average*, computed across the full test set, and *best-object performance*, computed on the single object where each model performs best.

### 4.3 Main Results

**General-Purpose MLLM Performance** Table 2 shows that general MLLMs have limited zero-shot assembly capability. For brick selection, stronger models achieve non-trivial but still unreliable performance: GPT-5.4 reaches the best overall accuracy, followed by Qwen-3.5-VL-27B. This suggests that current MLLMs can partially ground the target brick from visual context. The gap is larger for pose estimation: most models produce very large translation and rotation errors, indicating that these models lack robust 3D spatial reasoning for precise placement. As a result, strict step-level success remains near zero, with the best overall success rate only 0.36% and the best-object success rate only 4.44%. These results show that prompting alone is insufficient for reliable assembly. More analyses are in Appendix C.

**Brick-Composer Learning Performance** As shown in Table 3, the final Brick-Composer framework substantially improves both Gemma-3-12B and Qwen-3-VL-8B. Both models improve brick selection accuracy, greatly reduce translation and rotation errors, and achieve much higher step-wise success rates. On the best-object evaluation, the fine-tuned Qwen-3-VL-8B achieves near-perfect rotation prediction and a step-wise success rate of 42%. Its best-object mean translation error of 14.29 LDU is below one LEGO stud pitch (20 LDU); the overall strict step success is 14.27%. Figure 3 illustrates the resulting assembly; Figures 8, 9, 10, and 11 provide additional object-level examples, discussed in Appendix C.




Figure 3: Qualitative examples.

### 4.4 Ablation Studies

We conduct ablation studies to verify the effectiveness of each learning signal in our framework. As shown in Table 3, learning from designer supervision alone improves model performance despite limited data, especially for brick selection, while also reducing translation and rotation errors. In contrast, applying world feedback alone brings limited gains and can even hurt performance, as base models are not yet familiar with pose estimation and the additional feedback views introduce extra visual reasoning challenges. However, when combined with designer supervision, world feedback becomes effective and further improves upon the designer-supervised model. This benefit comes with additional inference cost: world feedback requires rendering erroneous states and prompting the model for another round of reasoning at each step. For a 70-step object, this can add roughly one hour of inference time. Finally, scaling with synthetic experience provides another substantial performance boost, pushing assembly capability to a new level. We leave a systematic study of scaling trends with data and model size to future work.

### 4.5 Non-MLLM Baselines and Robustness Analysis

**Visual Retrieval and CAD-Assisted Search** Table 4 compares visual retrieval and pose baselines; its MLLM rows use Qwen-3-VL-8B. We compare CLIP and DINOv2 retrieval using feature differences between before/after manual views, either over full images or aligned highlighted crops. Candidate similarities are averaged across six views; Table 7 gives the full retrieval comparison. Retrieval uses visual inputs only; CAD search additionally accesses the current assembly and selected-part geometry. We implement CAD render-and-compare search with pixel and DINOv2 losses, collision rejection, ten random restarts, and ten local refinement iterations. As shown in Table 4, the strongest retrieval baseline reaches 25.39% selection accuracy, versus 68.21% for Brick-Composer. CAD search reduces pose errors relative to prompted Qwen-3-VL-8B but remains behind the learned model, and is approximately 9.5× slower in the reported comparison. Table 8 additionally compares both trained backbones and the zero-pose baseline. Appendix D.1 distinguishes visual and CAD input assumptions, Appendix D.2 details retrieval, and Appendix D.3 specifies the losses, search budget, and runtime limitations. The full comparisons are collected in Appendix D. This local search is not an upper bound on geometric methods.

Table 4: Additional baselines. Selection (%), translation (LDU), and rotation (°); – denotes an unevaluated subtask.

| Approach | Sel. ↑ | Trans. ↓ | Rot. ↓ |
|---|---|---|---|
| CLIP (full image) | 13.24 | – | – |
| DINOv2 (full image) | 15.75 | – | – |
| CLIP (crop) | 21.70 | – | – |
| DINOv2 (crop) | 25.39 | – | – |
| Zero pose | – | 213.52 | 86.31 |
| CAD search | – | 187.33 | 60.48 |
| Qwen-3-VL-8B | 22.76 | 210.14 | 62.47 |
| Brick-Composer | **68.21** | **65.63** | **37.97** |

**Sensitivity to Tolerances** Figure 4(b) visualizes the tolerance analysis. Snapping translations to the standard stud grid and rotations to 90° orientations raises overall step success from 4.35% to 7.43% for Gemma-3-12B and from 14.27% to 21.20% for Qwen-3-VL-8B. This permissive diagnostic

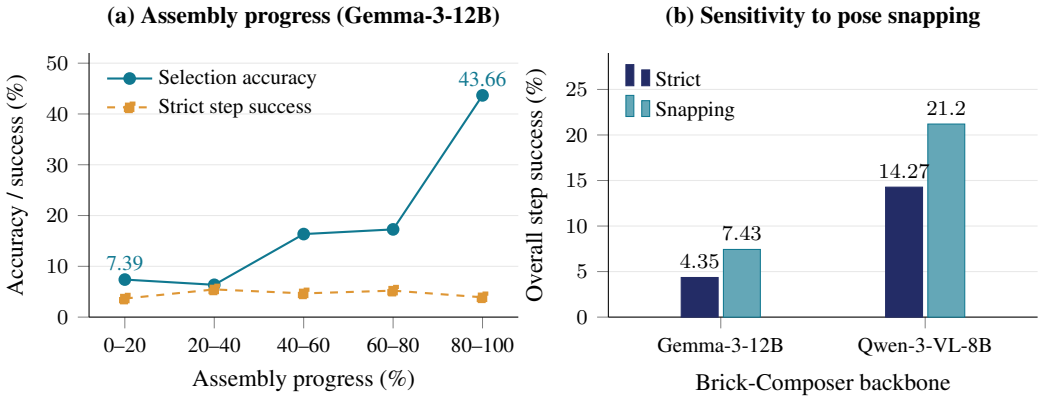


Figure 4: Additional experimental analyses from Tables 6 and 9. **(a)** Later steps improve selection for Gemma-3-12B, but strict step success remains low. **(b)** LEGO-specific pose snapping increases overall success for both learned backbones.

shows that some errors can be reduced by discretization, but half-stud offsets, SNOT connections, and continuously rotating parts violate its assumptions. Table 9 gives overall and best-object results; Appendix E distinguishes snapping from error thresholds. Strict evaluation remains primary.

**Assembly Progress and Full-Object Success** Figure 4(a) visualizes the progress analysis. For Gemma-3-12B, selection accuracy rises from 7.39% in the first fifth of assembly to 43.66% in the last fifth, consistent with a shrinking inventory; pose errors do not improve monotonically (Table 6; Appendix C). Full-object success is 0% for all evaluated methods. The 41.63% best-object result for Qwen-3-VL-8B measures successful steps, not completed objects. Evaluation with reference contexts diagnoses step-wise capability; predicted-history rollouts must additionally address exposure bias and compounding errors, as discussed in Appendix E.

## 5 Related Work

Our work connects spatial reasoning in MLLMs, language-guided brick construction, and robotic assembly. Prior studies improve spatial understanding through 3D grounding and spatial supervision (Hong et al., 2023; Chen et al., 2024), explore brick generation and assembly assistance (Pun et al., 2025; Kulits & Schmid, 2026; Huang et al., 2025), and learn assembly skills from manuals and geometric representations (Tie et al., 2025b; Jing et al., 2026). BC-Bench and Brick-Composer focus on visually grounded selection of diverse bricks and precise single-step pose estimation, connecting assembly evaluation with scalable learning from designs, feedback, and synthetic configurations. Appendix G provides the full discussion.

## 6 Conclusion

We formulate MLLM-based brick assembly as a sequential decision problem consisting of two coupled subtasks: brick selection and brick pose estimation. While our BC-Bench shows that general-purpose MLLMs remain limited in this setting, especially for precise pose prediction, we introduce **Brick-Composer**, a unified learning framework that adapts MLLMs through three complementary signals: human-designed assembly data, simulator-based world feedback, and scalable synthetic configurations. Our results show that Brick-Composer substantially improves assembly performance by increasing fine-grained brick selection accuracy, reducing translation and rotation errors, and raising strict step-level success. These results suggest that construction-oriented capabilities are learnable for large models when they are scaled with physically grounded supervision and feedback. Looking forward, our data provides a challenging yet meaningful testbed for studying how AI agents can move beyond perception toward compositional spatial reasoning and executable action in the real world. We hope our solution also encourages future research on large-scale assembly learning, spatially grounded model improvement, and scalable agents that can transfer from simulated construction environments to real-world robotics, manufacturing, and interactive design assistance.

## REFERENCES

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, 2025. URL https://arxiv.org/abs/2511.21631.

Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

Wenxiao Cai, Iaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. SpatialBot: Precise spatial understanding with vision language models. *arXiv preprint arXiv:2406.13642*, 2024.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded spatial reasoning in vision-language models. In *Advances in Neural Information Processing Systems*, 2024.

Erik Daxberger, Nina Wenzel, David Griffiths, Haiming Gang, Justin Lazarow, Gefen Kohavi, Kai Kang, Marcin Eichner, Yinfei Yang, Afshin Dehghan, and Peter Grasch. MM-Spatial: Exploring 3D spatial understanding in multimodal LLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3665–3671. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196714.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. In *Advances in Neural Information Processing Systems*, 2023.

Haochen Huang, Jiahuan Pei, Mohammad Aliannejadi, Xin Sun, Moonisa Ahsan, Chuang Yu, Zhaochun Ren, Pablo César, and Junxiao Wang. LEGO co-builder: Exploring fine-grained vision-language modeling for multimodal LEGO assembly assistants. *arXiv preprint arXiv:2507.05515*, 2025.

Zhi Jing, Jinbin Qiao, Ouyang Lu, Jicong Ao, Shuang Qiu, Yu-Gang Jiang, and Chenjia Bai. AssemLM: Spatial reasoning multimodal large language models for robotic assembly. *arXiv preprint arXiv:2604.08983*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025.

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

Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. MegaPose: 6D pose estimation of novel objects via render & compare. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 715–725, 2023. URL https://proceedings.mlr.press/v205/labbe23a.html.

Ruixuan Liu, Yifan Sun, and Changliu Liu. Robotic LEGO assembly and disassembly from human demonstration. *arXiv preprint arXiv:2305.15667*, 2023.

Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 515–524. PMLR, 2017.

OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, March 2026. Official OpenAI blog post. Accessed: 2026-05-26.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. DINOv2: Learning robust visual features without supervision, 2023. URL https://arxiv.org/abs/2304.07193.

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*, pp. 14798–14809, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026. Official Qwen blog post. Accessed: 2026-05-26.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Ishika Singh, Ankit Goyal, Stan Birchfield, Dieter Fox, Animesh Garg, and Valts Blukis. Og-vla: 3d-aware vision language action model via orthographic image generation. *arXiv preprint arXiv:2506.01196*, 2025.

Stefan Stevšić, Sammy Christen, and Otmar Hilliges. Learning to assemble: Estimating 6d poses for robotic object-object manipulation. *IEEE Robotics and Automation Letters*, 5(2):1159–1166, 2020.

Kexian Tang, Junyao Gao, Yanhong Zeng, Haodong Duan, Yanan Sun, Zhening Xing, Wenran Liu, Kaifeng Lyu, and Kai Chen. Lego-puzzles: How good are mllms at multi-step spatial reasoning? *arXiv preprint arXiv:2503.19990*, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy

Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Chenrui Tie, Shengxiang Sun, Yudi Lin, Yanbo Wang, Zhongrui Li, Zhouhan Zhong, Jinxuan Zhu, Yiman Pang, Haonan Chen, Junting Chen, Ruihai Wu, and Lin Shao. Manual2Skill++: Connector-aware general robotic assembly from instruction manuals via vision-language models. *arXiv preprint arXiv:2510.16344*, 2025a.

Chenrui Tie, Shengxiang Sun, Jinxuan Zhu, Yiwei Liu, Jingxiang Guo, Yue Hu, Haonan Chen, Junting Chen, Ruihai Wu, and Lin Shao. Manual2Skill: Learning to read manuals and acquire robotic skills for furniture assembly using vision-language models. In *Proceedings of Robotics: Science and Systems*, 2025b.

Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Chin-Yi Cheng, and Jiajun Wu. Translating a visual LEGO manual to a machine-executable plan. In *European Conference on Computer Vision*, pp. 677–694. Springer, 2022.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025. URL https://arxiv.org/abs/2508.18265.

Peiran Xu, Sudong Wang, Yao Zhu, Jianing Li, and Yunjian Zhang. SpatialBench: Benchmarking multimodal large language models for spatial cognition. *arXiv preprint arXiv:2511.21471*, 2025.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025a.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631. PMLR, 2025b.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

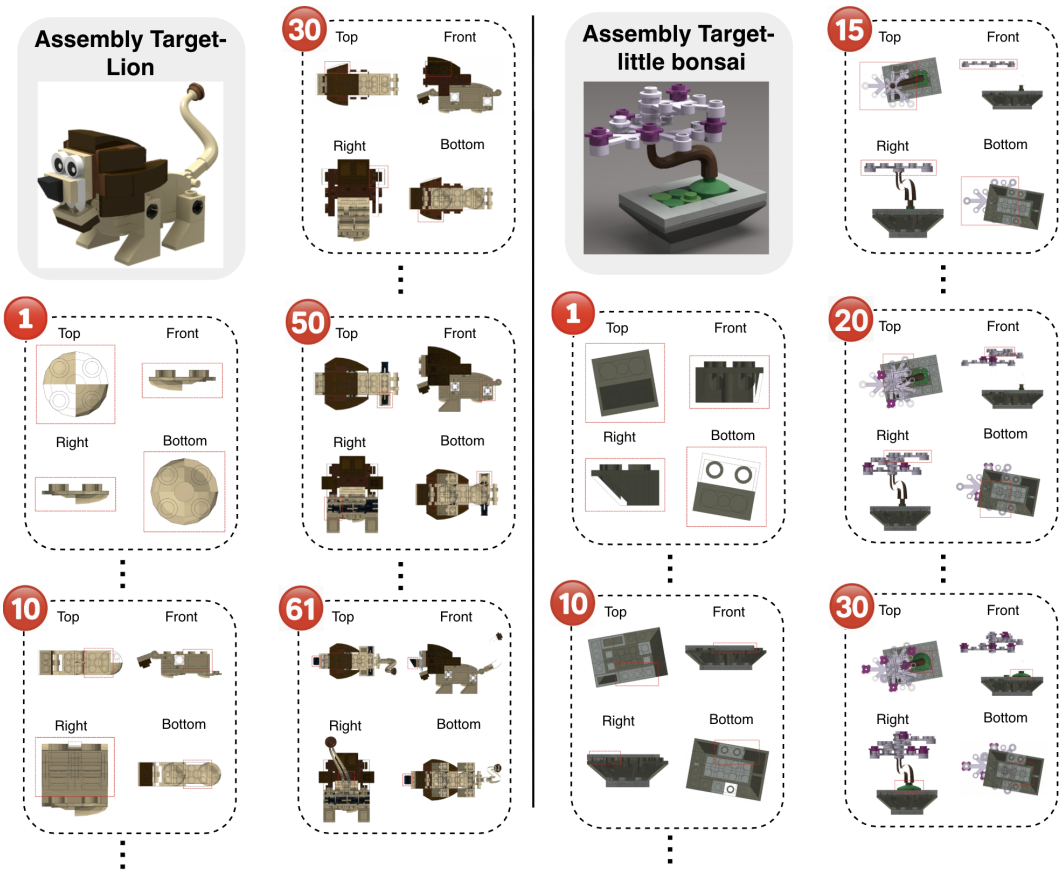


Figure 5: Examples of manual-style assembly sequences in BC-Bench. Each example shows a target LEGO-style object and selected construction steps rendered from multiple orthogonal views, with red boxes marking the newly added brick. BC-Bench provides six orthogonal views for each assembly step, along with annotations for brick identity and pose. The parts view data are shown in Figure 6.

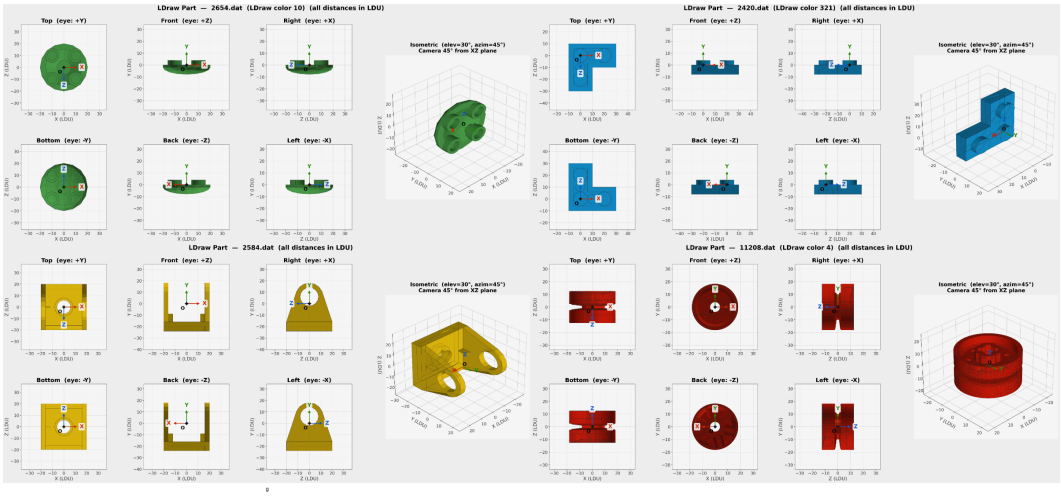

Figure 6: Examples of rendered part-level inputs in BC-Bench. Each part is visualized in its local coordinate system to expose its geometry, orientation, and scale.

## A Dataset for MLLM Assembly

**Visualization** Figure 1 illustrates the overall data formulation of BC-Bench, including the manual context, candidate bricks, current assembly state, and target pose annotation. To further clarify the

Table 5: Candidate-catalog statistics across assembly steps. Repetition counts refer to occurrences of the correct brick type in the remaining inventory. Percentages are reproduced at the reported precision and may not sum to exactly 100% due to rounding.

| **Catalog Size** | 1–10 | 11–20 | 21–30 | 31–40 | 41–50 | 51–60 | 61–80 | 81+ |
|---|---|---|---|---|---|---|---|---|
| Percentage (%) | 18.0 | 18.2 | 16.9 | 14.3 | 11.3 | 9.1 | 8.5 | 3.6 |

| **Target-Part Occurrences** | 1 | 2 | 3 | 4 | 5 | 6+ |
|---|---|---|---|---|---|---|
| Percentage (%) | 41.1 | 24.4 | 10.8 | 8.8 | 3.3 | 11.4 |

benchmark format, Figure 5 shows step-wise rendered assembly views, where each construction step is visualized from multiple viewpoints and the newly added brick is highlighted. Figure 6 provides additional examples of part-level inputs, showing the diverse brick geometries, colors, and metadata used for brick selection and pose estimation. Together, these visualizations demonstrate how BC-Bench converts human-designed objects into structured step-wise assembly tasks for MLLMs.

**Data Collection Details and Statistics** We collected 9,977 object files from online sources for research purposes and built our own simulator to render multi-view assembly states with step-wise annotations. We split the 82 pieces of training data at the object level, rather than at the step level, to avoid placing highly similar steps from the same object, and ensure the learning is conducted at the object level. This ensures that evaluation measures object-level assembly generalization rather than memorization of individual construction steps. Overall, BC-Bench contains 3,873 training steps from 82 objects. For simple and efficient evaluation, we selected 20 hard objects as core set and obtain 1,013 evaluation steps from the objects, covering a diverse set of LEGO-style objects and construction procedures.

**Candidate Inventory and Rendering** At step $t$, the catalog contains exactly the remaining bricks from the original object, including the brick required by the current instruction. Other candidates are parts needed at later steps, rather than separately sampled distractors. For $N_t$ candidates, the grid contains six columns and $\lceil N_t/6 \rceil$ rows, with a $120 \times 120$ pixel cell for each brick. Candidates are shuffled before rendering; real-object data use deterministic seeds, and all valid target-cell positions are recorded. Table 5 summarizes catalog sizes and repeated target parts across assembly steps. Approximately 59% of steps contain at least two interchangeable instances of the target brick type, with 3.04 occurrences on average. Set-based selection accuracy accepts any valid interchangeable instance, as defined in Section 4.

**Coverage and Generalization** The training and evalaution objects selected are with significant variation in part geometry, connection affordances, and placement configurations. The object-level split tests transfer to held-out designs rather than to additional steps from a training object. This diversity supports a diagnostic evaluation of step-wise assembly, although the number of objects remains limited and does not establish coverage of arbitrary construction tasks. Canonical part renderings provide a local reference frame; the target placement still generally requires a nonzero translation and a nonidentity rotation (Appendix D).

## B Trajectories for Multimodal Reasoning

We provide representative model interaction trajectories to illustrate our prompting design and task formulation for three settings: (1) brick selection, (2) brick pose estimation, and (3) world-feedback prompting, where models learn from simulator-rendered feedback. For readability, we omit the original images in these trajectories and replace each image position with the placeholder `<Image>` following the ShareGPT data format.

**Brick Selection Trajectories**

**"system"**: "You are a LEGO brick assembly expert. You will be given three images:

1. Original views: 6 orthogonal views (Top, Front, Right, Bottom, Back, Left) of the current LEGO assembly BEFORE the assembly step.

2. Target views: the same 6 orthogonal views AFTER the brick is placed, with the newly added brick highlighted with a red dashed bounding box.

3. Remaining brick catalog: a grid showing every brick that still needs to be placed (including the current step's brick), with part filenames (e.g. 3023.dat) labeled above each tile.

**Your task:** Compare the Original and Target views to identify which brick was placed, then locate it in the catalog. Output format — one line per brick: {part-filename}, row {R}, col {C}

**Rules:** {part-filename} must be the exact .dat filename shown in the catalog label. {R} and {C} are the 1-based row and column in the catalog grid (6 columns wide). If multiple bricks are placed in this step, output one line per brick in the same order as requested. Do NOT output any explanation, preamble, or extra text — only the formatted line(s).",

"conversations": [ { "from": "human", "value": "Below are the three inputs for this assembly step.

Image 1 Original views (brick not yet placed, 6 orthogonal angles: Top, Front, Right, Bottom, Back, Left): <image>

Image 2 Target views (brick placed, same 6 angles; newly added brick is highlighted with a red dashed bounding box): <image>

Image 3 Remaining brick catalog (rows and columns are 1-indexed from top-left, 6 columns wide; part names are labeled above each tile):<image> This step places exactly ONE brick.

Output a single line in the format: <part.dat>, row <R>, col <C>" }, { "from": "gpt", "value": "3035.dat, row 6, col 3" } ], }

**Brick Pose Estimation Trajectories**

{ **"system"**: "You are a LEGO brick assembly expert with precise knowledge of 3D coordinate systems and LDraw file formats. You will be given:

1. Original views: a composite of 6 orthogonal views (Top, Front, Right, Bottom, Back, Left) of the current LEGO assembly BEFORE the assembly step.

2. Target views: the same 6 orthogonal views of the assembly AFTER the brick is placed. The newly added brick is highlighted with a red dashed bounding box.

3. Part render(s): a rendered image of the brick (or bricks) to be placed, with the LDraw local coordinate frame annotated (red=+X, green=+Y, blue dot=+Z).

4. Current-state axes render: a 7-view composite (6 orthogonal + 1 isometric) of the assembly BEFORE this step, with LDU coordinate tick labels on every view and an isometric 3D projection in the right column. Use this to read off exact LDU coordinates and understand the spatial layout.

You will also be given text with part metadata (name, approximate physical size).

**Your task:** Determine the absolute global position (x, y, z in LDraw Units, LDU) and the rotation matrix that transforms the part's local frame into the global frame, for each brick being placed in this step. Coordinate system conventions (LDraw global frame): X axis : positive goes RIGHT in the standard front view. Y axis : positive goes DOWN (LDraw uses a left-handed, Y-down system). Z axis : positive goes TOWARD the viewer in the standard front view. 1 LDU

= 0.4 mm (one standard LEGO stud pitch = 20 LDU = 8 mm). Rotation matrix convention: The $3 \times 3$ matrix R encodes the orientation of the part. Each row gives the components of the brick's local axis in world space:

row 0 = brick local X axis expressed in world coordinates
row 1 = brick local Y axis expressed in world coordinates
row 2 = brick local Z axis expressed in world coordinates

**Output format** one block per brick (numbered Brick 1, Brick 2, when multiple): Brick <N>: Translation: x=<float>, y=<float>, z=<float> Rotation:

[<r00>, <r01>, <r02>]
[<r10>, <r11>, <r12>]
[<r20>, <r21>, <r22>]

**Rules:** Use decimal floats (e.g. 30.0, -50.0001, 11.1001). If there is only one brick in the step, still use "Brick 1:" as the header. Do NOT output any explanation, preamble, or extra text — only the formatted blocks above.",

"conversations": [ { "from": "human", "value": "Below are all inputs for this assembly step.

Image 1 Original views (assembly BEFORE this step, 6 orthogonal angles: Top, Front, Right, Bottom, Back, Left): <image>

Image 2 Target views (assembly AFTER this step; the newly placed brick is highlighted with a red dashed bounding box): <image>

Image 3 Part render: The brick (3035.dat (color 0)) The local LDraw coordinate frame is annotated on the image: red arrow = +X (right), green arrow = +Y (down), blue dot = +Z (toward viewer). <image>

Image 4 Current-state axes render (7 views: Top/Front/Right/Bottom/Back/Left + Isometric; light-grey background with LDU coordinate tick labels; use this to read off spatial positions and understand the 3D layout):<image>

Part information (brick to place this step):

Brick 1: 3035.dat | 3035 | dims ≈ 3.2 x 6.4 x 0.51 cm | LDraw color 0

Using all the images and part information above, predict the absolute global position (x, y, z in LDU) and rotation matrix for each of the brick listed above. Output exactly one 'Brick N:' block per brick, in the same order as listed." }, { "from": "gpt", "value": "Brick 1:

Translation:
x=0.0, y=-8.0, z=0.0

Rotation:
[1.0, 0.0, 0.0]
[0.0, 1.0, 0.0]
[0.0, 0.0, 1.0]"
} ], }

**Learning with Feedback Trajectories**

{ **"system":** "You are a LEGO brick assembly expert with precise knowledge of 3D coordinate systems and LDraw file formats. You will be given:

1. Original views: a composite of 6 orthogonal views (Back, Bottom, Left, Right, Up, Front) of the current LEGO assembly BEFORE the assembly step.

2. Target views: the same 6 orthogonal views of the assembly AFTER the brick is placed. The newly added brick is highlighted with a bounding box.

3. Part render(s): a rendered image of the brick (or bricks) to be placed, with the LDraw local coordinate frame annotated on the image.

4. Previous assembly render (omitted for the very first step): a composite rendering of the entire assembly from the immediately preceding step, giving you spatial context about the accumulated model.

5. Erroneous prediction render (when available): a rendering in which the incorrectly placed brick is shown in red, produced from a previous model prediction. Accompanying text specifies the wrong predicted rotation matrix and scalar error magnitudes (Euclidean translation error and geodesic rotation error) only — no absolute translation values or per-axis offsets are given. Treat this as a negative example — analyse where and how the prior prediction went wrong visually, then use that reasoning to arrive at the correct pose. You will also be given text with part metadata (name, approximate physical size).

**Your task:** Determine the absolute global position (x, y, z in LDraw Units, LDU) and the $3 \times 3$ rotation matrix that transforms the part's local frame into the global frame, for each brick being placed in this step. Coordinate system conventions (LDraw global frame):

• X axis : positive goes RIGHT in the standard front view.
• Y axis : positive goes DOWN (LDraw uses a left-handed, Y-down system).
• Z axis : positive goes TOWARD the viewer in the standard front view.
• 1 LDU = 0.4 mm (one standard LEGO stud pitch = 20 LDU = 8 mm). Rotation matrix convention: The $3 \times 3$ matrix R encodes the orientation of the part.

Each column gives the direction of one of the part's local axes expressed in the global frame: column 0 = direction of part's local X axis in global coordinates column 1 = direction of part's local Y axis in global coordinates column 2 = direction of part's local Z axis in global coordinates

Output format — one block per brick (numbered Brick 1, Brick 2, . . . when multiple): Brick <N>:

Translation:
x=<float>, y=<float>, z=<float>

Rotation:
[<r00>, <r01>, <r02>]
[<r10>, <r11>, <r12>]
[<r20>, <r21>, <r22>]

**Rules:**

• Use decimal floats (e.g. 30.0, -50.0001, 11.1001).
• If there is only one brick in the step, still use "Brick 1:" as the header.
• Do NOT output any explanation, preamble, or extra text — only the formatted blocks above.",
"conversations": [ { "from": "human", "value": "Below are all inputs for this assembly step.

Image 1 — Original views (assembly BEFORE this step, 6 orthogonal angles: Back, Bottom, Left, Right, Up, Front): <image>

Image 2 — Target views (assembly AFTER this step; the newly placed brick is highlighted with a bounding box): <image>

Image 3 — Part render: The brick (3023.dat — (color 71)) The local LDraw coordinate frame is annotated on the image: red arrow = +X (right), green arrow = +Y (down), blue dot = +Z (toward viewer)<image>

Image 4 — ERRONEOUS prior prediction (brick shown in RED — this is WRONG, do NOT copy it). A previous model attempt predicted the following incorrect pose for the brick(s) in this step. The image renders the brick(s) in red at the wrongly predicted location/orientation so you can see visually where the mistake is. Study the error and reason toward the correct answer.

Wrong prediction details: Brick 1 (3023.dat): Wrong predicted rotation matrix (row-major):
[1.0000, 0.0000, 0.0000]
[0.0000, 1.0000, 0.0000]
[0.0000, 0.0000, 1.0000]

Euclidean translation error : 23.7487 LDU Geodesic rotation error : 90.0000° <image>

Part information (brick to place this step): Brick 1: 3023.dat | 3023 | dims ≈ 0.8 x 1.6 x 0.51 cm | LDraw color 71. Using all the images and part information above, predict the absolute global position (x, y, z in LDU) and $3 \times 3$ rotation matrix for each of the brick listed above.

Output exactly one 'Brick N:' block per brick, in the same order as listed." }, { "from": "gpt", "value": "Brick 1:

Translation:
x=-10.0, y=-8.0, z=-20.0

Rotation:
[0.0, 0.0, -1.000001]
[0.0, 1.0, 0.0]
[1.000001, 0.0, 0.0]"
} ], }




Figure 7: Examples of synthesized assembly configurations used for synthetic experience learning. Each structure is generated by incrementally attaching sampled bricks through feasible connection positions, while filtering invalid placements based on collision avoidance and structural connectivity. A density preference further encourages compact layouts, allowing the synthesized data to provide diverse and scalable supervision for brick selection and pose estimation.

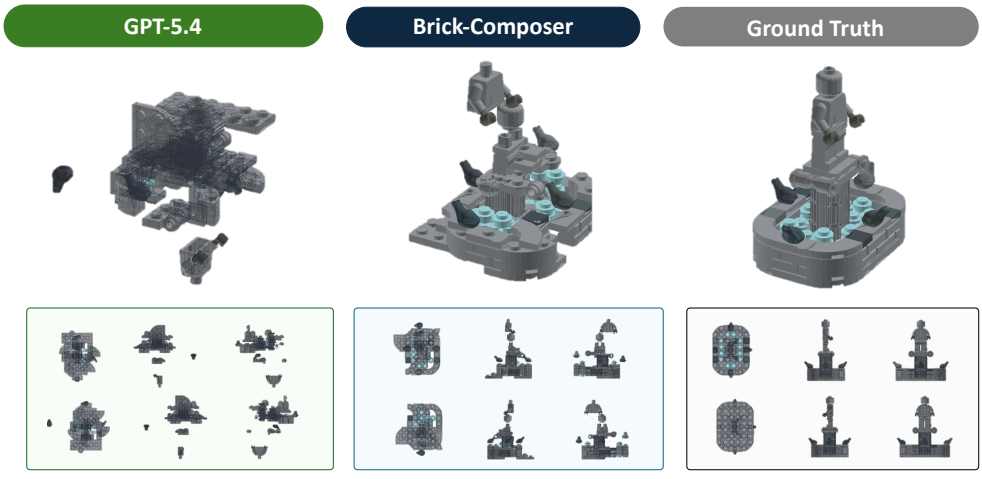


Figure 8: Additional qualitative examples of Brick-Composer assembly results. The generated assemblies show that the fine-tuned model can recover the overall structure and major components of target LEGO-style objects across multiple construction steps. These examples demonstrate the model's improved ability to jointly select appropriate bricks and place them into coherent object-level assemblies.

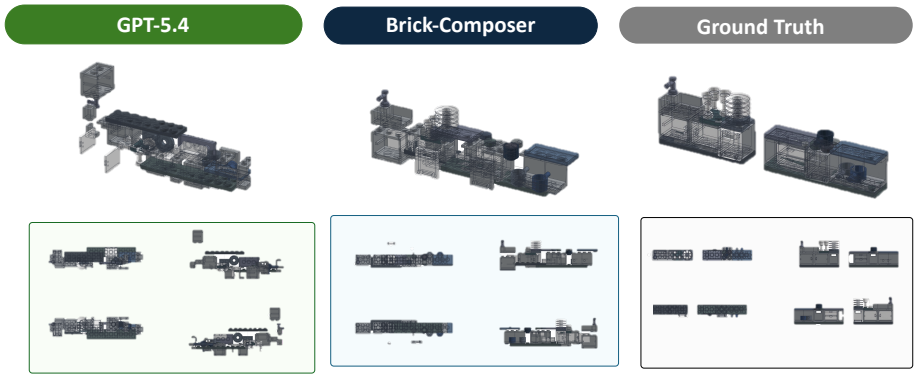


Figure 9: Additional qualitative examples of Brick-Composer assembly results. The generated assemblies show that the fine-tuned model can recover the overall structure and major components of target LEGO-style objects across multiple construction steps. These examples demonstrate the model's improved ability to jointly select appropriate bricks and place them into coherent object-level assemblies.

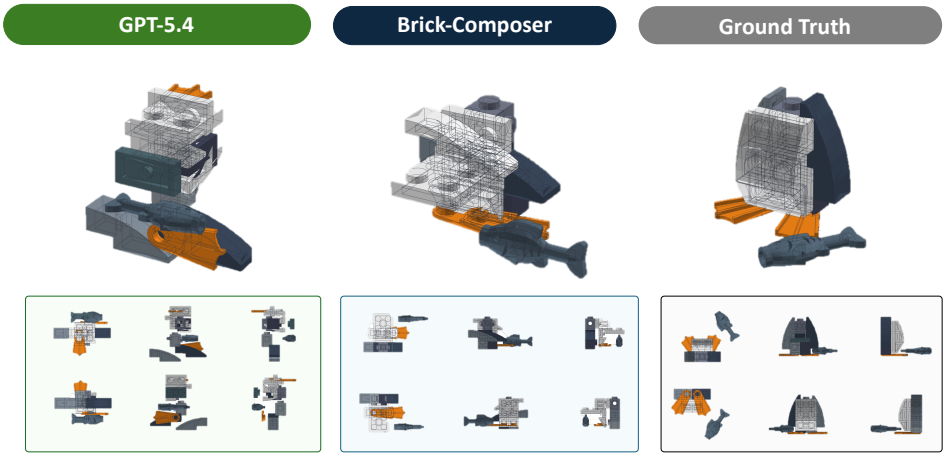


Figure 10: Additional qualitative examples of Brick-Composer assembly results.

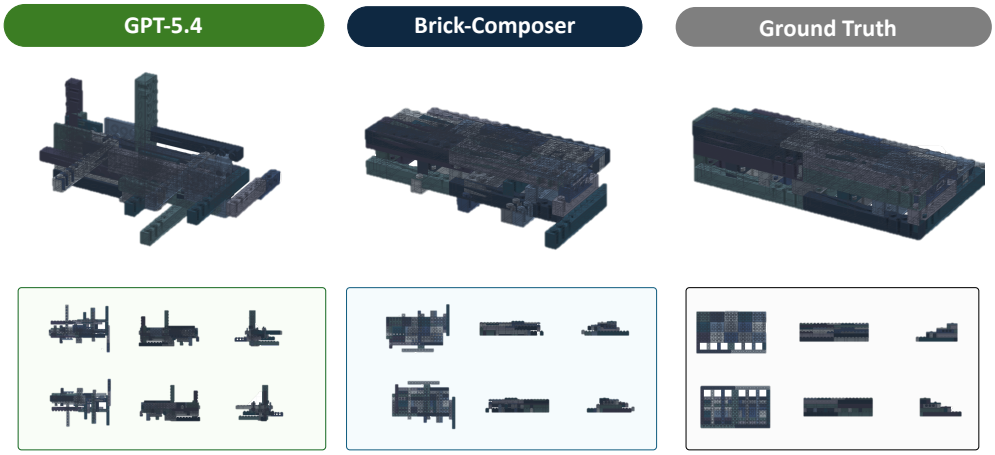


Figure 11: Additional qualitative examples of Brick-Composer assembly results.

Table 6: Performance by normalized assembly progress for Brick-Composer with Gemma-3-12B. The overall row reproduces the full-test-set result.

| Assembly Progress | Selection ↑ Acc. (%) | PE Trans. ↓ Err. (LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) |
|---|---|---|---|---|
| Overall | 17.95 | 123.69 | 45.43 | 4.35 |
| 0–20% | 7.39 | 126.10 | **41.36** | 3.65 |
| 20–40% | 6.36 | 124.47 | 42.09 | **5.45** |
| 40–60% | 16.35 | **119.43** | 51.22 | 4.67 |
| 60–80% | 17.27 | 125.54 | 52.26 | 5.21 |
| 80–100% | **43.66** | 122.91 | 44.78 | 3.90 |

# C More Experimental Analysis and Demos

**Step-Wise Brick Selection Complexity** We normalize each object's assembly sequence by progress and divide it into five intervals. Table 6 reports the resulting breakdown for Gemma-3-12B. Selection accuracy rises from $7.39\%$ in the first interval to $43.66\%$ in the final interval, consistent with a smaller remaining inventory at later steps. Pose errors do not improve monotonically with progress, and strict step-level success remains between $3.65\%$ and $5.45\%$. Thus, easier late-stage selection does not by itself resolve precise placement. These measurements characterize step-conditioned difficulty; they do not measure error accumulation in a rollout that feeds predicted states into later steps.

**Synthesized Examples** We provide additional visualizations of the synthetic experience data used in our learning framework. As shown in Figure 7, these examples are not copied from human-designed objects, but are procedurally generated by sampling bricks and attaching them through feasible connection positions. During generation, each accepted structure must satisfy basic physical constraints, including collision avoidance and structural connectivity. We further apply a density preference to encourage compact and meaningful assemblies rather than sparse or randomly scattered layouts. These synthesized examples expand the diversity of training configurations and provide scalable supervision beyond the limited set of human-designed objects.

**More Brick-Composer Qualitative Examples** We show additional qualitative examples of Brick-Composer assembly results in Figure 8, Figure 9, Figure 10, and Figure 11. These examples illustrate how the learned model can compose multi-step LEGO-style structures that visually resemble the target objects. While some fine-grained placement errors may still remain, the overall shapes and major structural components are successfully recovered, showing that our learning framework enables MLLMs to acquire non-trivial assembly capability beyond direct prompting.

# D Additional Baselines for Visual Assembly

## D.1 Input Assumptions and Related Approaches

BC-Bench evaluates the translation of visual assembly instructions into part choices and target poses. The manual specifies the desired next state visually; it does not directly provide the target transformation. Providing aligned, part-decomposed CAD states for both the current and next step would expose the added component and its transformation through state differencing, bypassing this inference problem. Part filenames appear in our prompt examples and catalog labels, but the visual task does not provide an aligned CAD decomposition of the target state.

Prior work on translating LEGO manuals into executable plans (Wang et al., 2022) and acquiring robotic skills from furniture manuals (Tie et al., 2025b) studies related interfaces. BC-Bench emphasizes selection from a remaining inventory of diverse parts and separately measures translation, rotation, and joint step success. BrickGPT (Pun et al., 2025) instead studies text-conditioned structure generation with simplified brick primitives, while BrickNet (Kulits & Schmid, 2026) studies brick-sequence generation. Their task definitions differ from manual-conditioned next-step assembly, so dataset size alone does not establish equivalent coverage.

Table 7: Training-free multiview retrieval baselines for brick selection. Highlight-guided cropping improves both backbones, but the best retrieval baseline remains below task-trained Brick-Composer.

| Method | Visual Backbone | Selection Acc. (%) ↑ |
|---|---|---|
| Full-image feature difference | CLIP | 13.24 |
| Full-image feature difference | DINOv2 | 15.75 |
| Highlight-guided crop | CLIP | 21.70 |
| Highlight-guided crop | DINOv2 | 25.39 |
| Direct prompting | Qwen-3-VL-8B | 22.76 |
| **Brick-Composer** | Qwen-3-VL-8B | **68.21** |

Geometric pose methods also make different input assumptions. ICP (Besl & McKay, 1992) registers 3D shapes, whereas the target next state here is specified by RGB manual images. CAD-assisted render-and-compare methods such as MegaPose (Labbé et al., 2023) estimate object poses using supplied object models, but do not by themselves select the intended component from an assembly inventory. We therefore evaluate visual-retrieval baselines and a custom CAD-assisted local-search baseline tailored to the available simulator. The latter is a budget-limited comparison, rather than a reproduction of ICP or MegaPose or an upper bound on geometric methods.

### D.2 Multiview Image Retrieval for Brick Selection

We split the visual catalog into individual grid cells and tightly crop each brick foreground, removing borders and excess background while retaining the original rendering. Each crop is one retrieval candidate. We evaluate CLIP (Radford et al., 2021) and DINOv2 (Oquab et al., 2023) without task-specific training under two variants. *Full-image feature-difference retrieval* subtracts the before-step image representation from the after-step representation for each of the six views. *Highlight-guided crop retrieval* uses the highlighted target region to crop the after-step image, excludes the highlight border, and applies the same crop coordinates to the before-step image before computing the feature difference.

Each candidate is compared with the change representation from each view. Similarity scores are averaged over the six views, and the highest-scoring candidate is selected. Both variants use the visual catalog and manual images without CAD models, part identifiers, or grid-to-CAD correspondences. Full-image retrieval retains global context but may dilute the signal of a small added part. Cropping makes that part more prominent, while introducing sensitivity to occlusion, shading, and crop boundaries. In both cases, a part embedded in an assembly must be matched to an isolated rendering with potentially different scale and orientation.

Table 7 shows that highlight-guided DINOv2 retrieval reaches $25.39\%$ accuracy, compared with $22.76\%$ for prompted Qwen-3-VL-8B and $68.21\%$ for its Brick-Composer variant. This supports the value of task-specific training for the tested assembly setting. It does not establish that retrieval or geometry-based approaches cannot benefit from stronger models or additional training. Highlight-guided retrieval also relies on annotations that may be unavailable in physical observations.

### D.3 Multiview CAD Render-and-Compare Pose Optimization

The CAD-assisted baseline receives the same multiview images and additionally accesses CAD geometry for the current assembly and selected brick. Each pose hypothesis is rendered using the same six orthogonal viewpoints and image resolution as the manual. We compare observed and rendered change images using two losses averaged across views: a pixel loss measuring the mean absolute RGB difference of normalized change images, and a feature loss measuring one minus the cosine similarity of their L2-normalized DINOv2 features. After bringing the terms to comparable ranges, the objective is

$$\mathcal{L}_{\text{CAD}} = \alpha \mathcal{L}_{\text{pixel}} + \beta \mathcal{L}_{\text{feature}}, \qquad \alpha = 1.0, \quad \beta = 0.25. \tag{10}$$

Hypotheses that intersect existing components are discarded, rather than assigned a soft collision penalty.

Table 8: Additional pose baselines and MLLM comparisons, reporting overall results. The zero-pose baseline predicts $T = (0, 0, 0)$ and $R = I$. The reported step-success figures for the auxiliary baselines are retained; their selection coupling is not specified in the available experiment description, so the pose-error columns provide the direct pose comparison.

| Approach | PE Trans. ↓ Err. (LDU) | PE Rot. ↓ Err. (°) | Step-Wise ↑ SR (%) |
|---|---|---|---|
| Zero translation, identity rotation | 213.52 | 86.31 | 0.09 |
| CAD render-and-compare | 187.33 | 60.48 | 0.18 |
| Gemma-3-12B prompting | 269.09 | 63.00 | 0.09 |
| Qwen-3-VL-8B prompting | 210.14 | 62.47 | 0.36 |
| Brick-Composer (Gemma-3-12B) | 123.69 | 45.43 | 4.35 |
| Brick-Composer (Qwen-3-VL-8B) | **65.63** | **37.97** | **14.27** |

We use ten random restarts. For each restart, translation is sampled uniformly from $[-100, 100]$ LDU along each axis relative to the assembly center, with resampling until the hypothesis lies inside the expanded assembly bounding box and avoids intersection. The initial rotation is the identity. Each refinement iteration evaluates the current pose and neighboring hypotheses obtained by translating in either direction along each axis and applying $30°$ rotational perturbations around the axes. Invalid hypotheses are removed, and the valid candidate with the lowest loss is retained. We run ten iterations per restart: four with a translation step of $10$ LDU, three with $5$ LDU, and three with $2$ LDU. The lowest-loss valid pose across restarts is returned.

The CAD baseline reduces mean translation and rotation errors relative to the two prompted MLLMs in Table 8, but does not exceed Qwen-3-VL-8B prompting in strict step success. It remains behind both trained Brick-Composer variants on these metrics. Repeated rendering also makes it approximately $9.5\times$ slower than Brick-Composer in the reported timing comparison. Its performance is limited by the local-search budget, initialization, and indirect image-matching objective; the timing ratio is specific to the measured implementation. The zero-pose baseline further shows that canonical part renderings do not make placement trivial: recovering the target transformation remains necessary.

## E Pose Tolerances, Symmetries, and Success Interpretation

**Physical Scale and Strict Evaluation** Translation is measured in LDraw Units (LDU): $1$ LDU is $0.4$ mm, and a standard LEGO stud pitch is $20$ LDU or $8$ mm. Thus, the best-object mean translation error of $14.29$ LDU for Qwen-3-VL-8B corresponds to approximately $5.72$ mm. This is a statement about the mean, not the fraction of predictions within one stud. The strict metric in Section 4 requires both a correct brick and an exact target pose, accounting for valid symmetries, without a task-level geometric tolerance.

**Symmetry Handling** For each brick, symmetry information is extracted from its `.dat` file and used to construct symmetry-equivalent ground-truth poses. The prediction is kept unchanged and compared against these equivalent references, taking the minimum error. This implements the symmetry-aware rotation metric in Section 4, so geometrically indistinguishable orientations are not penalized as different poses. Symmetry equivalence and tolerance relaxation are distinct: symmetry changes which reference orientations count as the same geometry, whereas snapping changes the prediction.

**Relaxed Snapping Analysis** We additionally evaluate a permissive post-processing setting that snaps predicted translations to the nearest standard stud position along each axis and orientations to the nearest $90°$ orientation. Nearest-grid translation snapping corresponds to a maximum adjustment of $10$ LDU per axis; the analogous one-axis angular intuition is $45°$. This is a snapping-based diagnostic, rather than a general success criterion defined by a Euclidean translation radius and an $SO(3)$ geodesic threshold. The pose errors below are measured after snapping, which is why they differ from the strict-setting errors.

Table 9 shows that overall step success rises from $4.35\%$ to $7.43\%$ for Gemma-3-12B and from $14.27\%$ to $21.20\%$ for Qwen-3-VL-8B. The corresponding best-object results rise to $26.70\%$ and

Table 9: Strict evaluation and relaxed snapping for Brick-Composer. Overall and best-object results are shown separately. Snapping is a permissive LEGO-specific analysis and does not imply physically valid placement for every part.

| Backbone | Setting | Overall Average | | | Best Object Performance | | |
|---|---|---|---|---|---|---|---|
| | | Trans. ↓ (LDU) | Rot. ↓ (°) | SR ↑ (%) | Trans. ↓ (LDU) | Rot. ↓ (°) | SR ↑ (%) |
| Gemma-3-12B | Strict | 123.69 | 45.43 | 4.35 | 24.63 | 12.86 | 18.75 |
| | Snapping | 110.51 | 32.84 | 7.43 | 17.22 | 7.74 | 26.70 |
| Qwen-3-VL-8B | Strict | 65.63 | 37.97 | 14.27 | 14.29 | 0.00 | 41.63 |
| | Snapping | **49.32** | **27.13** | **21.20** | **11.07** | **0.00** | **53.11** |

Table 10: Training and inference settings used for the learned models. Full-parameter adaptation applies to the language-model backbone; the vision encoder and multimodal projector remain frozen.

| Configuration | Value |
|---|---|
| Training framework / inference engine | LLaMA-Factory / vLLM |
| Trainable components | Full language-model backbone |
| Frozen components | Vision encoder and multimodal projector |
| Epochs | 1 |
| Optimizer | AdamW |
| Initial learning rate | $1 \times 10^{-5}$ |
| Learning-rate schedule | Cosine |
| Number of training GPUs | 4 |
| Per-device batch size | 1 |
| Gradient-accumulation steps | 2 |
| Global effective batch size | $4 \times 1 \times 2 = 8$ |
| Decoding | Greedy, temperature 0 |

53.11%. These gains indicate sensitivity to the evaluation convention, but the snapping assumptions are not universal: jumper plates permit half-stud offsets, studs-not-on-top (SNOT) structures use differently oriented connections, and some round parts allow continuous rotation. A downstream controller would require connection-specific tolerances. We therefore retain strict results as the primary comparison.

**Step-Level and Object-Level Performance** The reported full-object success rate is 0% for all evaluated methods. Step-level metrics expose improvements that this uniformly zero statistic cannot distinguish. For Qwen-3-VL-8B, the overall strict step success of 14.27% must be distinguished from the best-object value of 41.63%: the latter is the fraction of successful steps on a favorable object, not the fraction of objects completed. The evaluation uses supplied reference assembly contexts and primarily measures execution of individual manual-conditioned steps. It does not establish reliable autonomous construction with predicted history. Early mistakes can alter later observations and available connections, producing exposure bias and compounding failures. Evaluating predicted-state rollouts, recovery, and object completion remains necessary for long-horizon assembly.

## F Training, Inference, and Released Resources

**Image Inputs** The simulator produces six-view manual composites at $2515 \times 1770$ pixels, part renderings at approximately $3442 \times 1644$ pixels, and current-state renderings at $2482 \times 1179$ pixels. These are raw rendered dimensions, distinct from any subsequent model-specific image preprocessing. Candidate cells use the dimensions described in Appendix A.

**Optimization and Decoding** Table 10 gives the shared training settings. We use full-parameter fine-tuning of the language-model backbone rather than LoRA adaptation, while freezing the visual components. At inference, prompts specify the expected structured output, and predictions are

parsed according to the examples in Appendix B. The models generally follow these formats; format adherence should be distinguished from correctness of the predicted brick or pose.

**Role of Synthetic Experience** The manual provides the intended assembly sequence, so the learned skills are primarily visual part correspondence and pose inference rather than unconstrained design or global sequence planning. Synthetic configurations can train these skills through diverse local connections and spatial transformations even when the resulting structure lacks a recognizable object category. Table 3 compares designer supervision, feedback learning, and the full framework with synthetic experience. Synthetic configurations complement the semantic and affordance priors of human designs, but may not capture their aesthetics or higher-level construction logic. Generalization to manual-free planning is outside the present evaluation; an assembly and verification module could instead be combined with a separate planner in future work.

**Resource Coverage** The BC-Bench release comprises the 9977 object files, rendered images, simulator, annotations, object-level train/test splits, prompts, fine-tuning data, and training and inference configurations. They also include synthetic-data generation code and the large-scale synthetic dataset with the corresponding random seeds, distributed through the project repository and Hugging Face. Designer-originated assets and simulator-generated data have different provenance; access to a release should not be interpreted as unrestricted permission to redistribute every underlying design. Broader data curation requires attention to source-asset permissions.

**Transfer to Physical Assembly** Canonical part renderings supply a known local frame rather than randomly posed parts in a physical workspace. Deployment would additionally require part localization, camera and robot calibration, grasp selection, motion and contact control, and recovery from execution failures. Highlights and clean multiview renderings also simplify perception. The present results therefore establish improved simulated step-wise reasoning; physical transfer and reliable full-object construction remain open.

## G Related Work

**Spatial intelligence of MLLMs.** Recent work has studied spatial intelligence as a key capability of MLLMs. 3D-LLM (Hong et al., 2023) and SpatialVLMs (Chen et al., 2024) show that VLMs remain limited in spatial reasoning and improve this ability by injecting 3D scene data and large-scale spatial visual question answering data. Subsequent studies extend this direction to grounded region-level reasoning (Cheng et al., 2024), depth-aware spatial understanding (Cai et al., 2024), metric estimation and 3D grounding (Daxberger et al., 2025), video-based spatial intelligence (Yang et al., 2025a), embodied-agent evaluation (Yang et al., 2025b), and fine-grained spatial cognition benchmarks (Xu et al., 2025). These studies demonstrate the growing attention for spatial reasoning ability of MLLMs, motivating our work on assembly-specific reasoning with diverse bricks, where a model must identify the target brick and estimate its placement pose within an evolving structure.

**MLLMs for brick assembly and manufacturing.** MLLMs for assisted brick assembly and manufacturing have been studied only recently. BrickGPT (Pun et al., 2025) generates physically stable brick structures from text by fine-tuning a LLM, but it relies on simplified brick primitives, leaving open how models handle diverse parts with varied geometry and affordances. Recent LEGO-oriented work, including BrickNet (Kulits & Schmid, 2026) and LEGO Co-builder (Huang et al., 2025), expands the brick vocabulary and studies build-sequence generation or assembly-state analysis, but does not directly target visually grounded assembly assistance. AssemLM (Jing et al., 2026) further studies spatial MLLMs for robotic assembly, highlighting the importance of 3D geometric reasoning in assembly tasks. Our work is the first to address the lack of systematic evaluation of MLLMs for fine-grained brick selection and single-step pose estimation in realistic assembly settings by introducing a diverse benchmark and scalable learning pipeline for visually grounded brick assembly reasoning.

**Robotics for brick assembly.** Robotic assembly has long been studied as a challenging problem. Classical assembly settings require precise pose alignment and often rely on compliance control or contact feedback, while recent learning-based systems study assembly through manual understanding,

6D pose estimation, and robot policy learning. For example, Manual2Skill (Tie et al., 2025b) and Manual2Skill++ (Tie et al., 2025a) learn executable skills from manuals; robotic LEGO assembly has been studied from human demonstrations (Liu et al., 2023); pose-centric methods estimate 6D object or template poses from RGB-D or point-cloud observations (Deng et al., 2020; Stevšić et al., 2020); and recent VLA-style methods directly predict robot-specific grasp actions or action policies (Mahler & Goldberg, 2017; Zhen et al., 2024; Singh et al., 2025; Kim et al., 2025). Pose estimation and action execution remain core bottlenecks in robotic assembly. In contrast, our work opens a complementary angle for using MLLMs to select bricks and infer their target poses, providing an interpretable and simulation-evaluable step toward downstream robotic execution.

## AI USE STATEMENT

We used generative AI tools to polish the language of this manuscript. The authors take responsibility for its final content.