Brick-Composer: MLLMs Construct Everything from Building Blocks

Anonymous ACL submission

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-3-8B 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 and 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 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

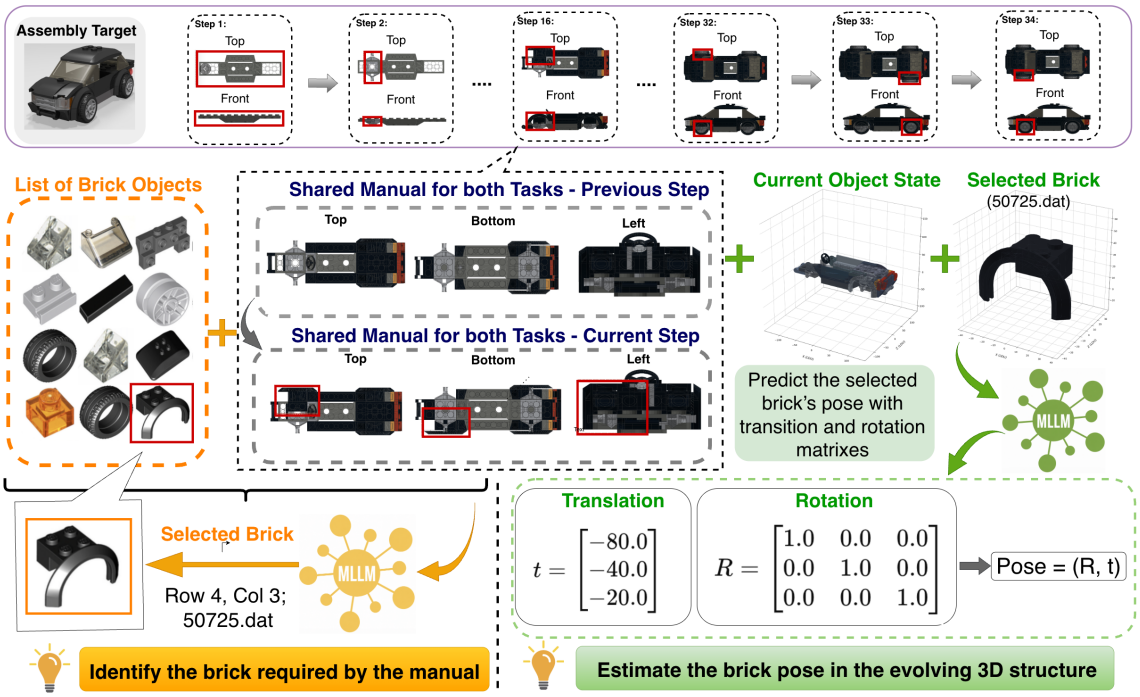

Figure 1: Overview of the BC-Bench task setting. **Left: Brick selection.** The model selects the required brick from a candidate grid using manual images. **Right: Brick pose estimation.** Given the manual context, current assembly state, and selected brick, the model predicts the brick's target pose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

views, candidate brick views, and evolving assembly states from orthogonal and isometric viewpoints.[1] 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 **Brick-Composer**, 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: 80 human-designed objects yield only about 3K step-wise training examples. *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 configurations. Empirically, Brick-Composer turns

[1]We respect designers' rights, follow responsible content-use practices, and avoid large-scale web scraping.

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 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 and 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 and 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.

3 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.

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 and 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	✓	✓	✓

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. We give more benchmark specific details in Appendix A.

3.1 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

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

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 θ predicts the grid coordinate of the brick required for the current step:

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

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 5.

3.2 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:

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

where $R_t^* \in SO(3)$ is the ground-truth rotation matrix and $T_t^* \in \mathbb{R}^3$ is the ground-truth transla-

tion vector. The translation vector and the rotation matrix are defined as:

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

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 θ predicts the absolute target pose:

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

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.

4 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. The complete prompt is provided in Appendix B. As shown in Table 2 and Section 5.3, 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.

4.1 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 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

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

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.

4.2 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:

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

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

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

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

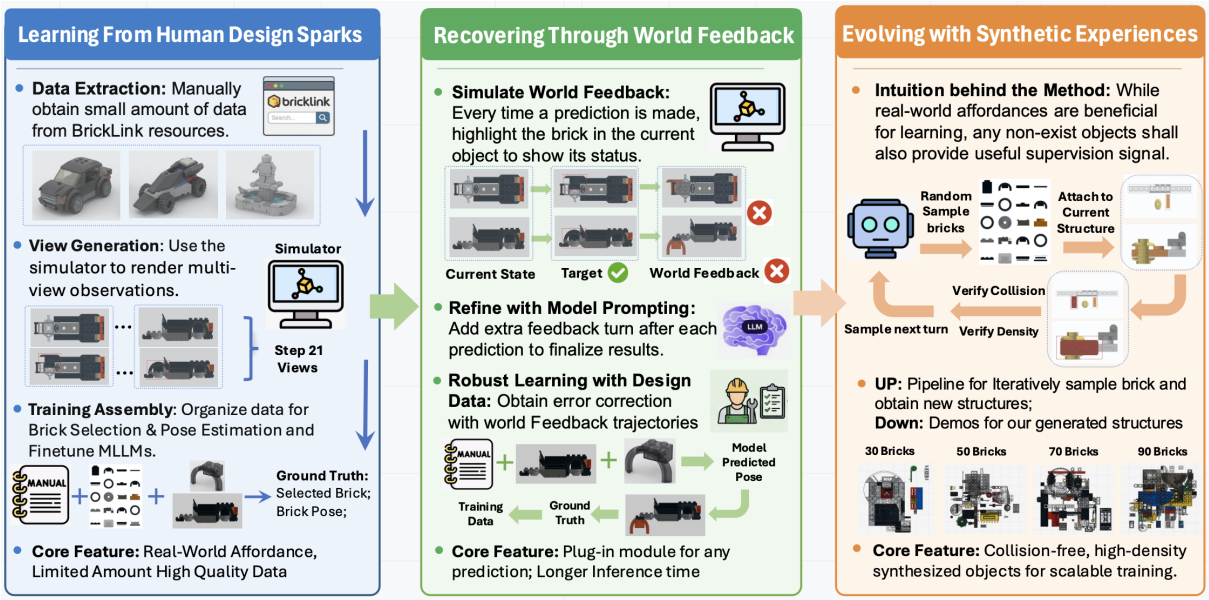

Figure 2: Overview of the Brick-Composer learning framework. We improve assembly reasoning through three complementary signals: Human design supervision, World feedback for error recovery, and Scalable synthetic objects for experience expansion. Together, they substantially enhance the model's assembly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

4.3 Scaling with Synthetic Experiences

Although human design 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. We use the same training objective as designer supervision. For each synthetic step t, the model receives the rendered visual input V_t^{arb}, task instruction I_t, and target answer A_t^{arb}:

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

5 Experiments

5.1 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.

5.2 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 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,

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

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 (%)
Performance Comparison of Learning Approaches for Model: 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**
Performance Comparison of Learning Approaches for Model: Qwen-3-8B-VL								
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**

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 $e^{\text{rot}} = \frac{1}{N}\sum_{t=1}^{N} \frac{180}{\pi}\theta_t^{\text{sym}}$. Accordingly, orientations that differ only by a valid brick symmetry are not penalized.

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.

5.3 Main Results

Limitation of General-Purpose MLLMs Table 2 shows that general MLLMs have limited zero-shot assembly capability. For brick selection,

Figure 3: Qualitative Examples of Model Assembly

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 2, 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 average translation error of 14.29 LDU is within one LEGO stud in most cases, indicating strong placement accuracy. Figure 3 shows a qualitative example where the assembled object visually resembles the target structure.

5.4 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.

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.

Limitations

Despite the well-constructed benchmark for evaluating MLLMs in assembly and the effectiveness of the Brick-Composer framework, this work has several limitations that also point to promising future directions. First, Brick-Composer studies brick assembly in simulation rather than direct real-world robotic execution. This choice allows us to isolate the core multimodal reasoning problem: whether MLLMs can understand assembly states, identify the correct component, and predict physically meaningful 3D placements under controlled and reproducible conditions. The simulator further provides accurate pose labels, consistent multi-view rendering, and scalable feedback, which are difficult to obtain at the same scale in real-world robotic settings. Nevertheless, transferring these capabilities to physical robots would require handling additional factors such as perception noise, occlusion, calibration errors, grasping constraints, contact dynamics, and execution failures. Thus, our results should be viewed as a strong step toward the spatial reasoning foundation needed for embodied assembly, rather than an end-to-end robotic assembly system.

Second, although human-designed assembly data provides realistic construction patterns and high-quality supervision, its scale is naturally constrained by the availability and copyright status of BrickLink-style design resources. This limitation motivates one of the key design choices of Brick-Composer: complementing human-designed data with simulator-generated arbitrary configurations. Such synthetic configurations allow us to scale physically valid supervision beyond the direct use of copyrighted designs, while still training the model on brick selection, connection geometry, and pose reasoning. At the same time, these configurations may not fully capture the semantic structure, aesthetic preference, and higher-level construction logic of designed objects. Future work could further explore copyright-aware data curation, procedurally generated object designs, and stronger methods for combining realistic human design priors with scalable synthetic experience.

Third, our evaluation focuses on step-wise assembly, where the model selects and places the next brick under a given assembly context. This setting is intentional: it provides a clear and measurable formulation for evaluating the two fundamental capabilities required by assembly, namely fine-grained brick selection and precise pose estimation. However, full object-level construction requires composing many such decisions over long horizons, where early mistakes can influence later steps. While Brick-Composer demonstrates that construction-oriented capabilities are learnable for MLLMs through physically grounded supervision and feedback, extending these gains to robust long-horizon autonomous construction remains an important direction for future work.

Ethical Statement on LLM Assistance

We primarily use GPT-5 as a tool for language refinement, including polishing text and improving clarity. All model-generated content is thoroughly reviewed and rewritten by human authors to ensure accuracy, originality, and adherence to research integrity standards.

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, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

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

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

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

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

Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. 2020. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on*

Robotics and Automation (ICRA), pages 3665–3671. IEEE.

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

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

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

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. 2025. 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*, pages 2679–2713. PMLR.

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

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

Jeffrey Mahler and Ken Goldberg. 2017. 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*, pages 515–524. PMLR.

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

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

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

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

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

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

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, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv: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. 2025a. Manual2Skill++: Connector-aware general robotic assembly from instruction manuals via vision-language models. *arXiv preprint arXiv:2510.16344*.

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

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

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, and 56 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

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

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2025a. 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*.

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. 2025b. 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*, pages 70576–70631. PMLR.

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

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 benchmark format, Figure 4 shows step-wise rendered assembly views, where each construction step is visualized from multiple viewpoints and the newly added brick is highlighted. Figure 5 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 To respect BrickLink designers' copyright, we do not conduct large-scale scraping or use large-scale designer data for training or evaluation. Instead, we manually downloaded 102 object files for research purposes and built our own simulator to render multi-view assembly states with step-wise annotations. We split the data at the object level with a $0.8 : 0.2$ train-test ratio, rather than at the step level, to avoid placing highly similar steps from the same object in both splits. 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 and 1,013 evaluation steps from 20 objects, covering a diverse set of LEGO-style objects and construction procedures.

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 2014 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 2014 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,

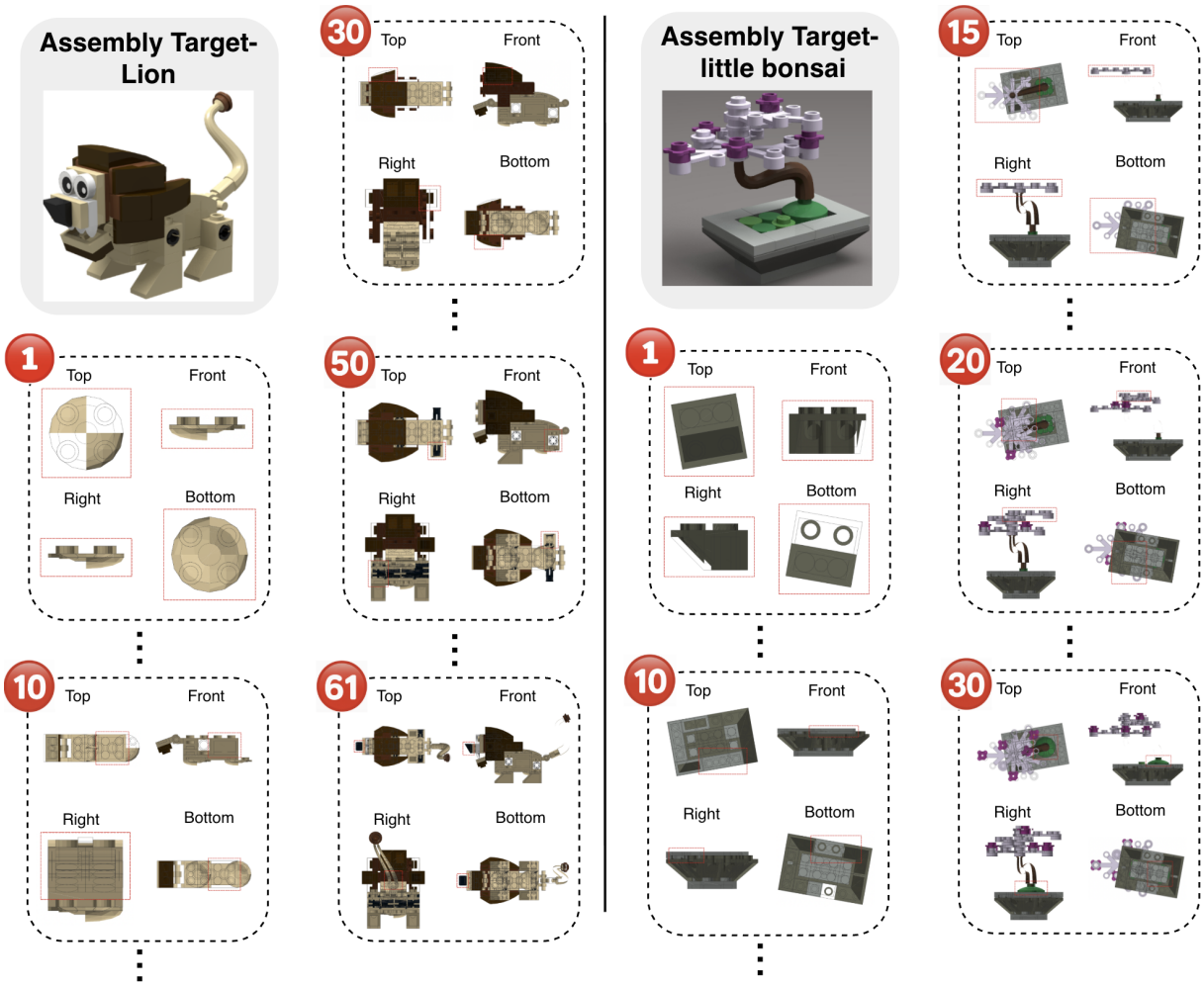

Figure 4: Examples of manual-style assembly sequences in BC-Bench. Each example shows a target LEGO-style object and selected construction steps rendered from multiple orthogonal views, with red boxes marking the newly added brick. BC-Bench provides six orthogonal views for each assembly step, along with annotations for brick identity and pose. The parts view data are shown in Figure **??**

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.

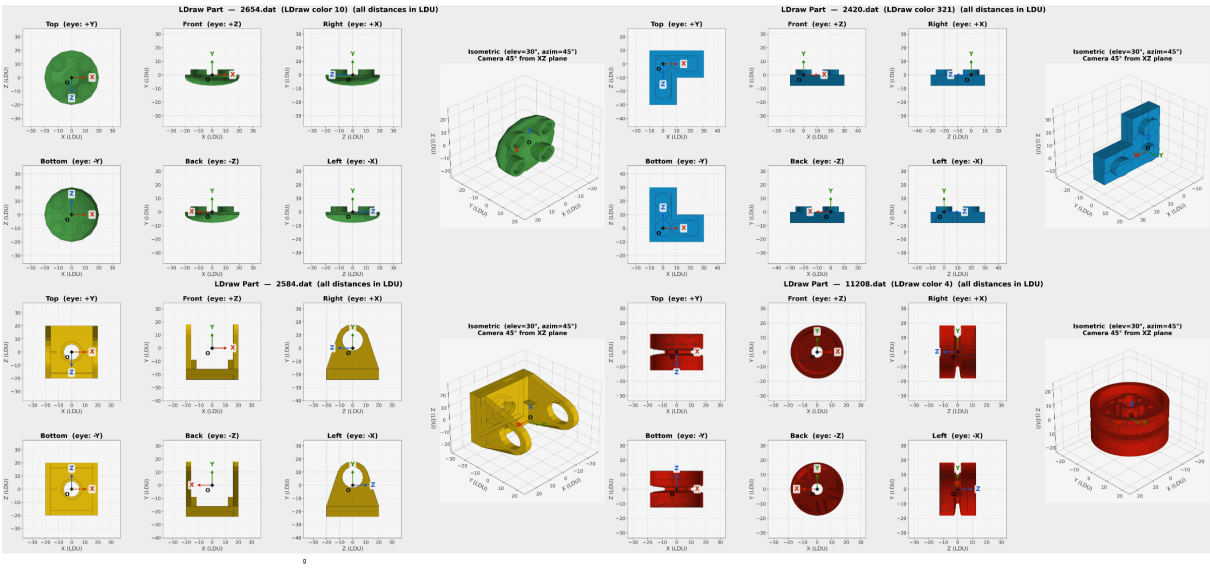

Figure 5: Examples of our rendered part-demo data in BC-Bench, we visualize the part within its own coordinate system, so that the model could have better knoweledge over their scale information.

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 300d73 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 2014 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 2248 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 3×3 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 3×3 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 3×3 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]"
}],

C More Experimental Analysis and Demos

Step-Wise Brick Selection Complexity We further analyze how brick selection difficulty changes across assembly steps. Our results show that selec-

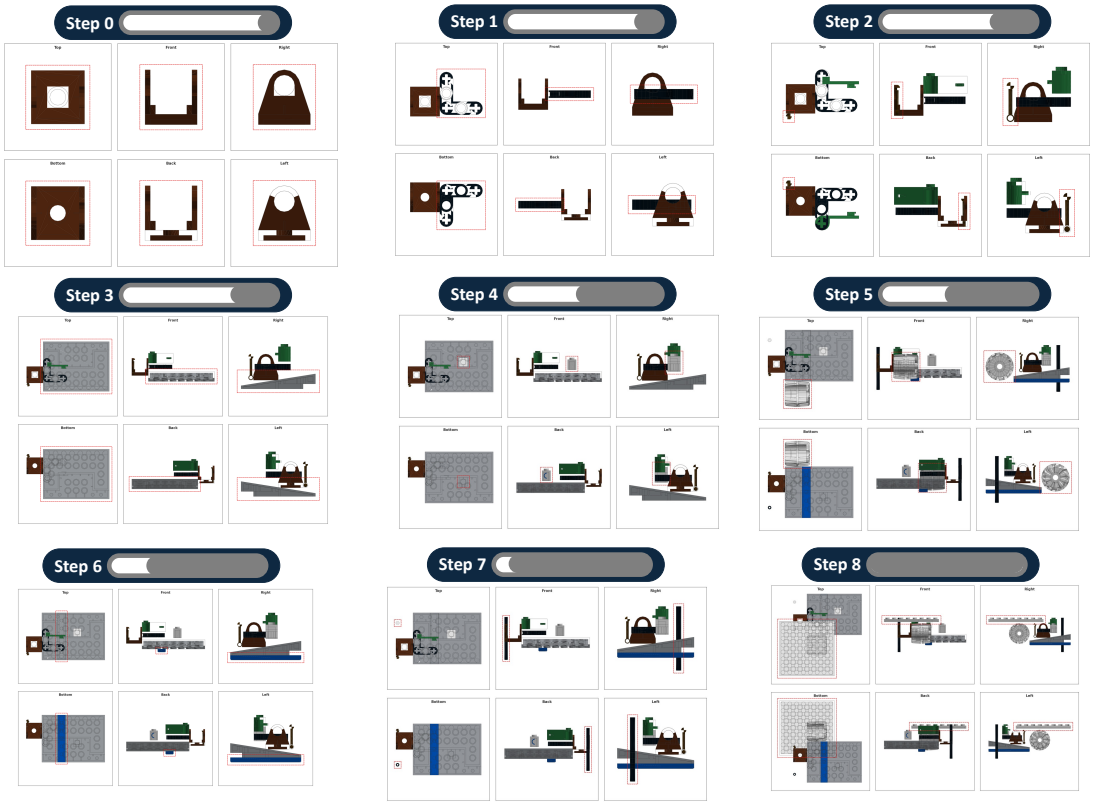

Figure 6: Examples of synthesized assembly configurations used for synthetic experience learning. Each structure is generated by incrementally attaching sampled bricks through feasible connection positions, while filtering invalid placements based on collision avoidance and structural connectivity. A density preference further encourages compact layouts, allowing the synthesized data to provide diverse and scalable supervision for brick selection and pose estimation.

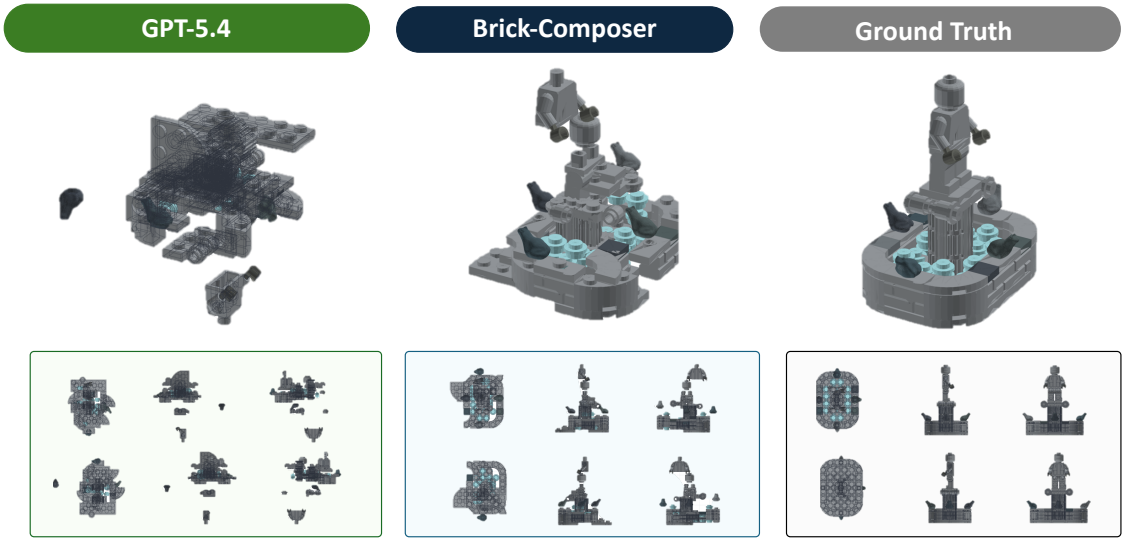

Figure 7: Additional qualitative examples of Brick-Composer assembly results. The generated assemblies show that the fine-tuned model can recover the overall structure and major components of target LEGO-style objects across multiple construction steps. These examples demonstrate the model's improved ability to jointly select appropriate bricks and place them into coherent object-level assemblies.

tion accuracy is generally lower in the early stage of assembly, especially within the first 80% of steps, while the last 20% of steps tend to be easier. This pattern suggests that early-stage assembly is more ambiguous: the partial structure is still sparse, the manual provides limited accumulated context, and many candidate bricks may appear visually similar or functionally interchangeable. In contrast, later steps often benefit from a more complete object structure, where the target placement region and required brick type become easier to infer from the accumulated assembly context. This finding reveals a critical challenge for real-world assembly agents. Errors made in early brick selection are not isolated; selecting the wrong brick can change the subsequent structure, misalign later pose predictions, and trigger cascading failures throughout the remaining assembly process. Therefore, improving brick selection is not merely a preliminary localization problem, but a core requirement for reliable long-horizon assembly. In this sense, robust brick selection is as important as pose estimation, since accurate placement is only meaningful when the correct component has first been identified.

Synthesized Examples We provide additional visualizations of the synthetic experience data used in our learning framework. As shown in Figure 6, 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 7. 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.