ASSEMBLYVERSE: TOWARDS MULTIMODAL 3D ASSEMBLY REASONING & PLANNING

Anonymous authors
Paper under double-blind review

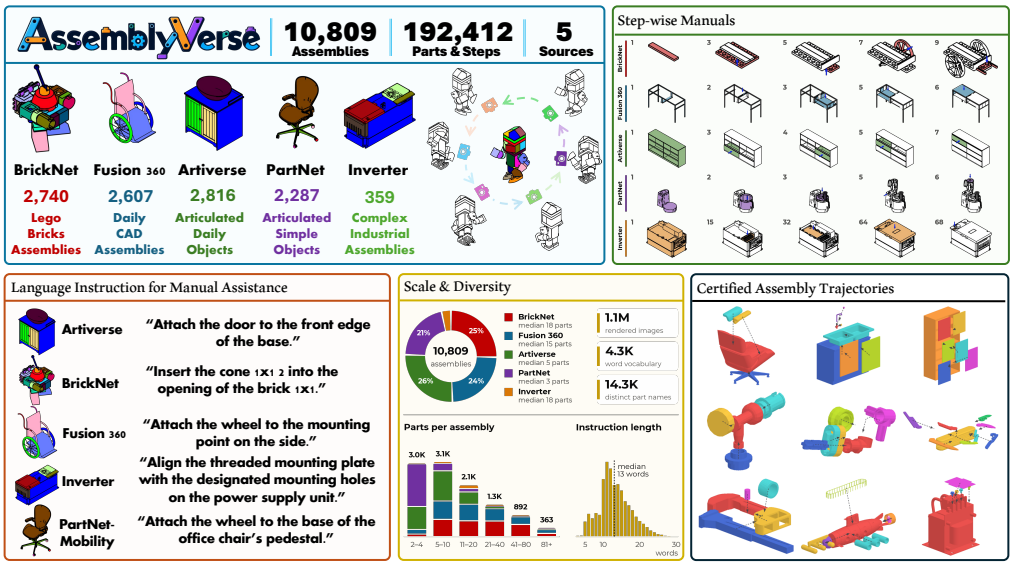


Figure 1: **AssemblyVerse**, a large-scale multimodal process-level 3D assembly dataset. AssemblyVerse incorporates objects from five diverse source domains, each with different part statistics, into one large-scale dataset under a unified format. For each object in the dataset, AssemblyVerse includes a step-wise multimodal assembly manual with both line drawings and text instructions, the ground-truth 3D final pose of each part in the assembly, and certified ground-truth assembly trajectories. The part that is added in each step of the step-wise manuals is highlighted in the line drawing.

ABSTRACT

The task of 3D assembly is not merely predicting a sequence of independent part poses; it is a long-horizon, state-conditioned decision problem that couples reasoning about *what* to assemble next with planning *how* to move each part into place. Existing benchmarks address either 6-DoF pose reasoning or trajectory prediction in specialized regimes, leaving open the general process-level learning across domains and initial configurations. Towards this end, we introduce **AssemblyVerse**, a large-scale, multi-domain dataset for process-level multimodal assembly reasoning and planning of diverse objects and complex assembly configurations. AssemblyVerse contains 10,809 assemblies spanning five domains and 192,412 assembly steps. Each step is aligned with rendered instruction diagrams, text instructions, evolving assembly states, and geometrically verified collision-free trajectories. Building on AssemblyVerse, we present **AssemblyMind**, a vision-language 3D assembly model that grounds each subsequent assembly step through an auxiliary order-grounding head and autoregressively generates 6-DoF trajectories as reordered discrete pose tokens. Our *staging-randomized supervised fine-tuning* improves robustness to diverse initial part configurations, while **GV-GRPO** directly optimizes geometric validity of predicted trajectories without sacrificing final-pose accuracy. Extensive experiments across domains demonstrate that strong performance on AssemblyVerse enables transferability of process-level learning. Compared to the previous state of the art, AssemblyMind substantially improves order grounding, final-pose prediction, and collision-free trajectory generation.

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.

1 INTRODUCTION

3D assembly is a fundamental task for embodied intelligence that brings together perception, spatial reasoning, and geometry-grounded planning in applications such as manufacturing, maintenance, repair, and assembly assistance. Successful assembly requires more than recognizing individual parts; an agent must identify the next part from the current state, infer its target placement from multimodal instructions, and execute a collision-free trajectory. Instruction manuals provide a natural high-level specification of this process, but they do not directly ground each instruction in the evolving 3D assembly state nor translate the specified placement into a physically executable trajectory.

Existing resources address only certain aspects of these challenges. Large-scale 3D datasets describe static part structure (Mo et al., 2019) or articulation (Xiang et al., 2020), while manual-grounded assembly methods primarily focus on part ordering and final-pose prediction (Wang et al., 2022b; Zhang et al., 2025; Jing et al., 2026). More recently, AssemblyBench (Li et al., 2026) introduced assembly trajectories for procedurally generated industrial objects. However, existing datasets rarely combine complete assembly procedures, multimodal instructions, and collision-validated trajectories across diverse and complex assemblies, limiting cross-domain generalization.

We introduce **AssemblyVerse**, a large-scale dataset and benchmark for process-level multimodal 3D assembly reasoning and planning, spanning diverse object categories and assembly procedures with varying complexity. AssemblyVerse contains 10,809 assemblies across five domains and 192K assembly steps, each paired with rendered instruction diagrams, natural-language instructions, part meshes, and a geometrically verified assembly trajectory. Unlike annotations that capture only assembly order or the final poses of parts, our geometry-grounded framework aligns semantic instructions with executable motion, tightly coupling "what goes where" with how it gets there. AssemblyVerse therefore provides dense, physically grounded supervision that can help models learn to reason about an evolving assembly and plan feasible part trajectories.

To make use of this supervision, we introduce **AssemblyMind**, a VLM-based model that formulates 3D assembly as state-conditioned grounding and trajectory planning. Given multimodal instructions, part geometry, a starting pose, and the current assembly state, AssemblyMind predicts both which part to assemble next and how to place it. An auxiliary *order-prediction head* grounds the assembly sequence, while an autoregressive VLM generates a start-conditioned 6-DoF trajectory as discrete pose tokens. We further introduce *pose-token reordering*, predicting the final pose before intermediate waypoints to provide an explicit goal and reduce autoregressive error accumulation.

To improve robustness to diverse initial configurations, we train with *staging-randomized supervised fine tuning (SR-SFT)*, which randomizes the starting poses of parts during training. Beyond imitation learning, we introduce **geometry-verified group relative policy optimization (GV-GRPO)**, which uses geometric feasibility checks as reinforcement signals to encourage task-consistent, collision-free trajectories.

Empirical results show that AssemblyMind predicts accurate final poses and geometrically feasible trajectories on the AssemblyVerse dataset across diverse starting conditions, object categories, and assembly complexity. Moreover, AssemblyMind transfers well across assembly domains and datasets; this robust zero-shot performance demonstrates its wide applicability.

In this work, we make the following main contributions:

- *AssemblyVerse*, a large-scale multimodal dataset and benchmark for 3D assembly spanning 5 domains and varying complexity, which pairs 10,809 assemblies and 192K assembly steps with instruction diagrams, text instructions, target poses, and geometrically verified trajectories.
- *AssemblyMind*, an autoregressive VLM-based framework for multimodal 3D assembly that predicts accurate final poses and initial-pose-conditioned collision-aware trajectories through discrete pose-token reordering, with an auxiliary order-prediction head for part-order grounding.
- Staging-randomized SFT and geometrically-verified GRPO for training AssemblyMind, enabling robust, geometrically feasible trajectory generation from diverse starting poses without sacrificing final-pose accuracy.
- AssemblyMind achieves state-of-the-art performance, improving trajectory feasibility and final-pose accuracy while generalizing across assembly categories, complexity levels, and datasets. These results demonstrate that *a single model can jointly achieve precise placement and reliable motion planning for general 3D assembly*.

2 Related Work

3D Assembly Datasets. Part-level shape datasets such as PartNet (Mo et al., 2019) provide fine-grained part decompositions and have been turned into part-assembly benchmarks that predict the 6-DoF pose of every part from its geometry (Zhan et al., 2020). Later datasets extend geometric assembly to fractured objects (Sellán et al., 2022), to functional pairwise assembly of everyday objects (Qi et al., 2025), and to joints between CAD parts in the Fusion 360 Gallery assembly dataset (Willis et al., 2022). These datasets supervise final poses or pairwise mating but not the trajectories that produce them. Assembly trajectory has instead been studied in planning: Assemble Them All (Tian et al., 2022) plans physically plausible assembly trajectory and sequences for industrial assemblies through disassembly in simulation, and ASAP (Tian et al., 2024) extends this to sequence planning for robotic assembly with physical feasibility. These planners operate on one object at a time with full geometric knowledge and do not pair the resulting procedures with instructions. The closest dataset to ours is AssemblyBench (Li et al., 2026), which provides multimodal manuals, part models, and assembly trajectories for 2,789 synthetic industrial objects and evaluates them in a physics simulator. Our AssemblyVerse shares the same data format but draws its assemblies from a substantially broader set of five assembly domains, including articulated objects (Iliash et al., 2026; Xiang et al., 2020), LEGO sets (Kulits & Schmid, 2026) and CAD assemblies (Willis et al., 2022). AssemblyVerse pairs every step with multi-view diagrams and a text instruction, and every trajectory is certified using a geometric checker.

Manual-Guided 3D Assembly. IKEA-Manual (Wang et al., 2022b) pairs IKEA furniture manuals with part-level 3D models and poses, and IKEA Manuals at Work (Liu et al., 2024a) grounds manual steps in internet assembly videos. MEPNet translates LEGO manuals into machine-executable plans through 2D keypoint detection and 2D-3D projection (Wang et al., 2022a). Manual-PA (Zhang et al., 2025) aligns parts with their illustrations to predict the assembly order and the 6D pose of each part, AssemLM (Jing et al., 2026) predicts 6D assembly poses as discrete tokens of a multimodal language model with a point-cloud encoder, and Manual2Skill (Tie et al., 2025) uses a vision-language model to extract hierarchical assembly graphs from manuals for robotic furniture assembly. AssemblyDyno (Li et al., 2026) goes further and jointly predicts the assembly order and the trajectory of every part. These methods predict which part is assembled and where it goes, and in the case of AssemblyDyno how it gets there, but always from the dataset's own starting configuration. Building on the tokenized pose interface of AssemLM, our AssemblyMind model instead predicts the full trajectory conditioned on an observed intial pose that may lie anywhere around the assembly space, not just the dataset's own starting configuration.

3D-Aware Vision-Language Models. Vision-Language Models have been extended to the domain of 3D understanding and generation by encoding point clouds directly (Xu et al., 2024), by lifting multi-view image features into 3D (Hong et al., 2023; Zhou & Tai, 2026a), and by building embodied generalist agents that perceive, reason, and act in 3D scenes (Huang et al., 2023; Zhou & Tai, 2026b). SpatialVLM (Chen et al., 2024; Cai et al., 2026) instead learns spatial reasoning from large-scale, automatically generated spatial question answering. Most of these models answer in natural language. However, assembly requires numerically precise geometric outputs, which AssemLM and AssemblyMind obtain through discrete pose tokens and rotation-equivariant point features (Deng et al., 2021). Reinforcement learning with verifiable rewards, in particular GRPO (Shao et al., 2024) and its variants Dr. GRPO (Liu et al., 2024b) and DAPO (Yu et al., 2026), have been applied to vision-language models for visual perception (Shen et al., 2025) and manipulation reasoning (Song et al., 2026), with rewards that check answer correctness, localization, or agreement with a reference trajectory. In contrast, our GV-GRPO verifies the whole predicted trajectory against the existing geometry, so that a valid trajectory that differ from the reference is rewarded as well. We leverage GV-GRPO to improve the performance of AssemblyMind.

3 The AssemblyVerse Dataset

AssemblyVerse is a multi-domain, multimodal, process-level 3D assembly dataset that uses the per-model organization of AssemblyBench (Li et al., 2026) but contains 10,809 assemblies with 192,412 steps from a wide variety of domains: articulated everyday objects (ArtiVerse (Iliash et al., 2026), PartNet-Mobility (Mo et al., 2019)), LEGO sets (BrickNet (Kulits & Schmid, 2026)), CAD assemblies (Fusion 360 Gallery (Willis et al., 2022)), and complex industrial 3D assemblies (inverters,

which are devices for controlling industrial motors). See Table 1 for the comparison with other datasets. Every assembly comes with part meshes and the ground-truth part assembly order, and every step comes with a multi-view rendered instruction diagram, natural language (text) instruction, the ground-truth final pose of the current part, and a certified trajectory of the current part in the form of $K{=}12$ key poses (the 12th key pose is the final pose of the part in the complete assembly).

3.1 Dataset Generation Pipeline

After each data source is converted to per-part meshes in a shared frame (Appendix B), two stages produce the data (Figure 2): a disassembly planner that yields the part order and certified trajectories, and a rendering and language instruction stage that generates the assembly instruction manual.

(1) Validate-then-confirm disassembly planning. Ground-truth trajectories are obtained by disassembling the object, one part per round, then reversing the result. Each round ranks the parts that could come off at the current step leveraging heuristic geometric priors (e.g., leaf-first, outside-in), proposes candidate trajectories for each[1], and certifies it by sweeping the part against every part still present. A trajectory is accepted only if the part leaves without new interference. A VLM (Qwen3-VL-8B (Bai et al., 2025)) then picks the most sensible next part among the *certified* candidates only, so that semantics shape the order but never the validity.

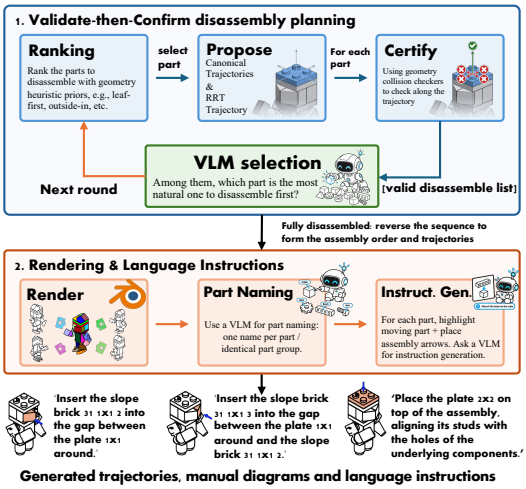


Figure 2: We use a validate-then-confirm disassembly planning with a geometric engine driven, and VLM selection manner; then we render the manual diagrams, followed by VLM-guided part naming and instruction generation.

(2) Rendering & Language Instructions. Blender (Blender Online Community, 2025) renders every assembly step by step, following the fixed camera and diagram style proposed in AssemblyBench Li et al. (2026), with a segmentation pass that locates every part in every view. The VLM is shown the target part highlighted both in the corresponding assembly step and in the completed assembly, and is instructed to avoid color-based descriptions, viewpoint-dependent terms, and names already assigned to other part groups. A multimodal model (Qwen3.5-9B (Team, 2026)) then writes one sentence per step from the diagram in which the moving part is highlighted, named, and drawn with its trajectory arrow, describing intrinsic relations rather than viewpoint-dependent ones. For steps in which the target part is not visible in any view, we generate the instruction from the part name and its mesh-verified mating partners and mark such cases explicitly in the dataset annotations.

3.2 Task Setting and Metrics

Task setting. Our task setting differs from previous works Li et al. (2026); Zhang et al. (2025) in that we use autoregressive assembly prediction and start-conditioned assembly generation for the task of 3D assembly: *(i) Autoregressive prediction.* Previous works (Zhang et al., 2025; Li et al., 2026) predict all the final poses / assembly trajectories for all steps all at once. In our setting, however, the assembly is predicted sequentially following the manual order, with each previously placed part retained at its *predicted* assembled pose in the input to the next prediction step, thereby enabling the model to handle longer assembly sequences. *(ii) Start conditions.* Every assembly is predicted under two start conditions: the part's starting pose (*GT init*), and starting from anywhere in the 3D space. To evaluate the second start condition, we report the mean performance of the six canonical start poses around the current assembly (*free-space*).

Metrics. We evaluate the 3D assembly process using the following metrics. A part's final pose *passes* if the Chamfer Distance (CD) between its point cloud at the predicted pose and at the ground-

[1] We use template-based extraction for screws and pins and canonical linear trajectories when applicable. If these trajectories fail collision certification, we use an RRT-based planner to find a collision-free alternative. See App. A for more details.

Table 1: **Comparison with 3D assembly datasets.** *Traj.*: a per-part trajectory, physics-simulated (^{sim}), certified collision-free (^{cert}) or from video (^{vid}). *Parts*$_{avg}$: mean parts per assembly.

Dataset	Domain	#Asm.	Parts$_{\text{avg}}$	Text	Order	Pose	Traj.
3D assembly without instructions							
Fusion 360 Asm. (Willis et al., 2022)	CAD	8,251	18.7	✗	✗	joints	✗
Two by Two (Qi et al., 2025)	Daily objects	517 pairs	2	✗	✗	✓	✗
Manual-grounded 3D assembly							
IKEA-Manual (Wang et al., 2022b)	Furniture	102	10.4	✗	✓	✓	✗
IKEA Manuals at Work (Liu et al., 2024a)	Furniture	36	7	✗	✗	✓	✓^{vid}
MEPNet (Wang et al., 2022a)	LEGO	8,000	–	✗	✓	✓	✗
Manual-PA (Zhang et al., 2025)	Furniture	6,871	–	✗	✓	✓	✗
AssemBench (Jing et al., 2026)	Furniture	150K steps	–	✓	✓	✓	✗
AssemblyBench (Li et al., 2026)	Industrial	2,789	6.7	✓	✓	✓	✓^{sim}
AssemblyVerse (ours)	**Diverse**	**10,809 (192K steps)**	**17.8**	✓	✓	✓	✓^{cert}

truth final pose is at most 0.01, following (Li et al., 2026). Its path is *valid* if, at every probe along the predicted polyline, the part's clearance c to the placed parts satisfies $c \geq \tau$ in free space and $c \geq \max(0, c^{\star}(d) - \delta)$ where the certified path is itself in contact at the same distance d to the seat. We report: (i) **Part Accuracy (PA)**, the fraction of parts with a correct final pose, averaged across assemblies; (ii) **Success Rate (SR)**, the fraction of assemblies in which all parts achieve correct final poses; (iii) **Trajectory Validity Rate (TVR)**, the fraction of parts whose trajectories are valid and whose final poses are correct; and (iv) **End-to-End Success Rate (E2E)**, the fraction of assemblies in which every part satisfies the TVR criterion. On AssemblyBench, the predicted trajectories are additionally executed in the Newton physics simulator, which reports PA$_{sim}$ and SR$_{sim}$ as well.

4 Proposed Method: AssemblyMind

4.1 Overview

An assembly is an unordered set of N rigid parts $\{P_1, \ldots, P_N\}$, each represented as a point cloud, together with a manual, consisting of diagrams $M_1, \ldots, M_N$, and a natural language instruction l_t for each step. We denote the instructional difference image (Zhang et al., 2025) at step t as $D_t = |M_t - M_{t-1}|$, for $t > 1$ and $D_1 = M_1$. We build AssemblyMind on Qwen3-VL (Bai et al., 2025) and use the same multimodal backbone for both part grounding and trajectory generation.

Order grounding. The manual specifies the sequence of assembly steps, but not which physical part each step refers to. AssemblyMind scores every manual step against every part's point cloud and reads the assembly order off a one-to-one assignment using a lightweight auxiliary *order grounding head*, which grounds the overall order of all parts before each step's assembly begins.

Trajectory prediction. Let $\mathcal{S}_t$ be the parts already placed at their final poses and let $p_1 \in SE(3)$ be the *starting pose* at which the grounded part $P_{\sigma(t)}$ is presented. The model predicts

$$f_\theta\big(M_{t-1}, D_t, l_t,\ P_{\sigma(t)}, \mathcal{S}_t, p_1\big) \longmapsto (p_2, \ldots, p_K), \qquad p_k \in SE(3), \tag{1}$$

a sequence of key poses ending at the final pose p_K. The starting pose is a task *condition* rather than a target: different starting configurations may lead to the same final assembly. At inference, grounding and trajectory generation alternate, and each part is placed at its predicted final pose before processing the next step.

Inputs and encoders. A step is represented by the diagram M_{t-1} before the operation, the difference image D_t, and the instruction l_t. The two images, M_{t-1}, D_t, are encoded by the vision tower of the VLM base while the instruction is processed by its language model. The selected part and the current partial assembly are encoded by a shared equivariant point-cloud encoder (Deng et al., 2021; Wang et al., 2019), followed by a learned projection into the VLM embedding space. Together with the starting-pose tokens, these representations form the multimodal input to the model.

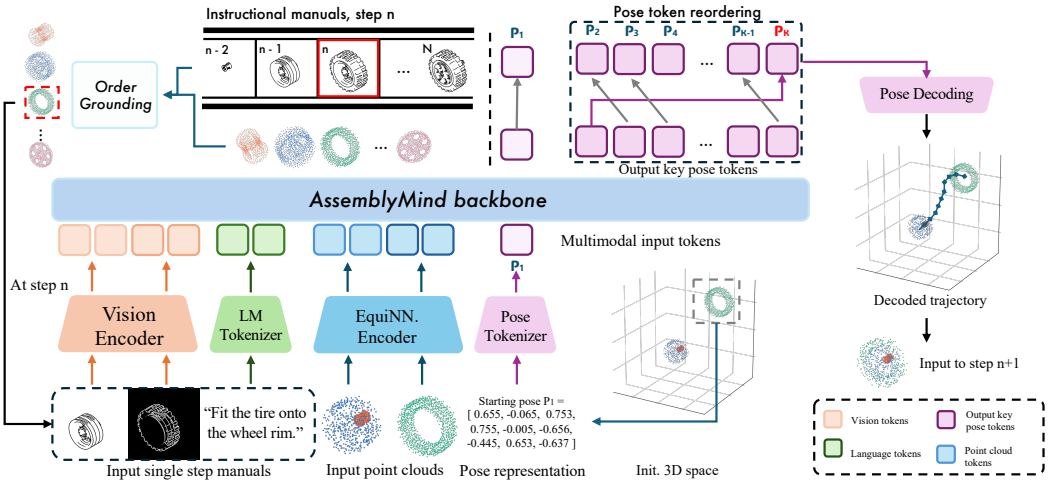


Figure 3: **AssemblyMind overview (one assembly step).** At step n, the manual instruction, the part and partial-assembly point clouds, and the starting pose p_1 are tokenized and compose the input tokens. Since we perform pose token reordering, the output tokens are reordered back, decoded into $SE(3)$ key poses, and the part placed at p_K updates the assembly passed to step $n{+}1$.

4.2 Trajectory Tokenization

Pose and Trajectory representation. Each key pose is encoded in the normalized assembly frame as a 9D vector $\mathbf{p}_k = [\mathbf{r}_k \mid \mathbf{t}_k] \in \mathbb{R}^9$, where $\mathbf{t}_k \in \mathbb{R}^3$ is the displacement of the part center from the world origin and $\mathbf{r}_k = [\mathbf{a}_k \mid \mathbf{b}_k] \in \mathbb{R}^6$ is a continuous 6D encoding of a rotation $\mathbf{R}_k \in SO(3)$ using the first two rows of $\mathbf{R}_k$; predicted vectors $\mathbf{a}_k$ and $\mathbf{b}_k$ are orthonormalized using Gram–Schmidt, and their cross product defines the third column, yielding a valid right-handed rotation with $\det(\mathbf{R}_k) = +1$. This representation avoids the discontinuities of Euler angles and the double cover of quaternions (Zhou et al., 2019). To tokenize the poses, every dimension is uniformly quantized into B bins over $[-1, 1]$, each bin a dedicated token, so a trajectory of K key poses is a sequence of $K \times 9$ pose tokens. Discrete prediction allows the model to represent uncertainty over multiple plausible pose values without directly averaging them in continuous coordinate space.

Pose token reordering. Since under the natural token order $p_1, p_2, \ldots, p_K$, the model must extrapolate—i.e., every intermediate pose is predicted before the endpoint it is meant to reach—an early error propagates into the final pose through the autoregressive inference process. We therefore reorder the sequence of prediction so that the final pose directly follows the given starting pose:

$$\big[\underbrace{p_1}_{\text{cond}},\ p_K,\ p_2, p_3, \ldots, p_{K-1}\big], \tag{2}$$

which conditions intermediate trajectory generation on both the observed starting pose and an explicit predicted goal. The 9 tokens of p_1 are supplied as a prefix and never predicted, so the model predicts $(K-1)\times 9$ tokens (within a pose, rotation tokens precede translation tokens). Training minimizes the cross-entropy error over the predicted tokens, with those of the final pose upweighted, since an inaccurate final pose invalidates the assembly step, whereas small errors in intermediate poses are often less consequential.

4.3 Order Grounding

The manual gives the sequence of steps but not which 3D part each step refers to. We recover this with a lightweight head g_ϕ on the frozen features of the trained AssemblyMind model. Since the full manual is available beforehand, the head scores all steps against all parts jointly:

$$g_\phi\big(\{D_t, M_t, l_t\}_{t=1}^{N}, \{P_i\}_{i=1}^{N}\big) \longmapsto S \in \mathbb{R}^{N\times N}, \tag{3}$$

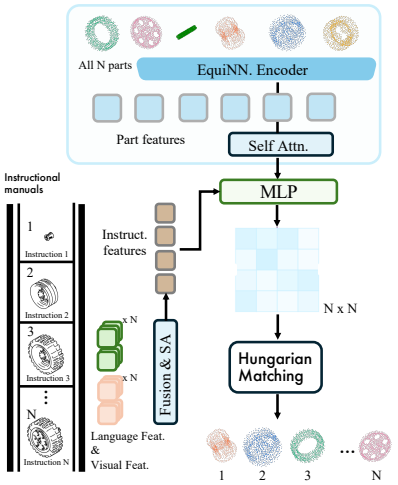


Figure 4: Overview of our order-grounding head.

where ϕ represents the learnable parameters, trained using ground truth assembly order, and $\Sigma(N)$ is the set of permutations of the N parts. During inference the order is the assignment $\sigma^\star = \arg\max_{\sigma\in\Sigma(N)} \sum_t S_{t,\sigma(t)}$, solved by Hungarian matching. Thus, using equation 3, we assume the part order for a given assembly is inferred before the subsequent stages begin. In the order grounding head, we first extract instructional and part features, apply self-attention over the part features, and then use gated fusion to integrate the full set of multimodal features. The similarity of each candidate pair is subsequently scored using a simple multi-layer perceptron (MLP) (see Figure 4). More details are in Appendix A.

Since parts can be identical, let $\mathcal{G}(t)$ denote the parts identical to the reference part of step t. We minimize a group-tied cross-entropy in both directions:

$$\mathcal{L}_{\mathrm{G}} = -\tfrac{1}{2}\Big[\frac{1}{T}\Sigma_t \log \sum_{i\in\mathcal{G}(t)} \mathrm{softmax}_i\, S_{t,\cdot} + \frac{1}{I}\Sigma_i \log \sum_{t:\, i\in\mathcal{G}(t)} \mathrm{softmax}_t\, S_{\cdot,i}\Big]. \tag{4}$$

Since both softmax operations ranging over all N parts, no ground-truth prefix enters training. Only the head parameters are updated, in a short phase after the main trajectory model training.

4.4 Training

Supervised Fine-Tuning Cold Start. We first fine-tune AssemblyMind, with multimodal context and starting-pose tokens provided as conditions, to predict the remaining pose tokens. Starting poses are arbitrary in deployment, since a part may be handed to the robot anywhere around the assembly. We therefore propose *staging randomization*: for every training sample, it is re-drawn on every pass, mixing the reference start with local perturbations and free-space starting poses sampled around the assembly. For each randomized starting pose, we construct a collision-free connection to the reference start while preserving the original certified local trajectories. The resulting trajectory is then resampled to the same fixed number K of key poses.

Geometry-Verified GRPO. Supervised training treats every deviation from the reference trajectory alike, whether it collides or not. The same deterministic geometric verifier used to certify AssemblyVerse can also evaluate sampled trajectories, allowing us to use trajectory validity directly as a reward. The policy π_θ adds LoRA adapters (Hu et al., 2021) to the attention projections of the language model, leaving the adapter-free network as the reference policy π_{ref}. We use training samples drawn under the same staging randomization as prompts, while for each prompt the starting pose tokens are forced and G trajectories $\{\tau_i\}_{i=1}^{G}$ are sampled from the pose vocabulary at temperature T.

Reward Signal. A rollout is verified exactly as in evaluation and scored by a binary reward:

$$r(\tau_i) = \mathrm{final}(\tau_i) \times \mathrm{valid}(\tau_i),\ r(\tau_i) \in \{0, 1\}. \tag{5}$$

Here: $\mathrm{final}(\tau_i) = \mathbb{1}[\mathrm{CD}(\mathcal{P}(p_K), \mathcal{P}(p^\star)) < \delta_{\mathrm{final}}]$, while $\mathrm{valid}(\tau_i)$ indicates whether the densely interpolated trajectory is collision-free with the already assembled geometry.

Training Objective. For each prompt, we sample G trajectories and compute group-relative advantages following Dr. GRPO (Liu et al., 2024b), using mean-centering without within-group standard-deviation normalization:

$$A_i = r(\tau_i) - \frac{1}{G}\sum_{j=1}^{G} r(\tau_j). \tag{6}$$

We optimize the policy using a clipped GRPO objective with KL regularization:

$$\mathcal{L}_{\mathrm{GV}} = -\mathbb{E}_{i,s}\left[\min(\rho_{i,s}A_i,\ \mathrm{clip}(\rho_{i,s}, 1-\epsilon, 1+\epsilon)A_i)\right] + \beta\,\mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})\,, \tag{7}$$

where $\rho_{i,s} = \pi_\theta(a_{i,s} \mid \cdot)/\pi_{\mathrm{old}}(a_{i,s} \mid \cdot)$. Following DAPO (Yu et al., 2026), groups in which all sampled trajectories receive the same reward are skipped, since there is no relative learning signal.

5 Experiments

Implementation Details. We employ Qwen3-VL 2B (Bai et al., 2025) as our VLM base model. We train on AssemblyVerse training set and evaluate on its test split and on other benchmarks with

Table 2: **Trajectory prediction** results on the AssemblyVerse test set. *Free-space*: see Section 3.2. ‡: trained with start-conditioning on the same dataset. †: SFT trained with point cloud encoders. The vision-language models are evaluated with the same orders as ours in 'predicted order'.

	Ground-truth order									Predicted order							
	Reference start				Free-space start					Reference start				Free-space start			
Method	PA	SR	TVR	E2E	PA	SR	TVR	E2E	KD	PA	SR	TVR	E2E	PA	SR	TVR	E2E
Vision-language models																	
Qwen3-VL-2B†	71.92	33.17	54.95	9.67	12.97	1.71	3.12	0.00	–	41.71	11.85	32.12	5.49	5.87	2.37	1.97	0.00
Qwen3.8 27B	22.10	1.99	15.58	1.42	5.81	0.75	1.61	0.00	–	12.12	0.94	5.81	0.00	2.08	0.19	0.12	0.00
Specialized models																	
Manual-PA‡	14.69	3.70	3.97	0.76	14.69	3.70	3.79	1.53	0.721	14.23	3.60	3.85	0.76	14.23	3.60	1.48	0.00
AssemblyDyno‡	40.80	9.29	26.75	4.64	7.38	2.97	3.79	1.53	0.732	40.44	8.91	26.81	5.02	6.74	0.36	2.63	0.18
AssemblyDyno‡ + SA	29.00	5.02	10.83	1.23	10.53	2.76	2.79	0.58		28.73	4.45	11.01	1.33	15.06	1.61	5.02	0.24
Ours																	
AssemblyMind (SR-SFT)	94.18	67.79	75.44	27.29	**86.34**	54.69	49.81	14.78	**0.786**	73.17	41.71	59.42	20.95	67.71	35.50	39.73	12.10
AssemblyMind (GV-GRPO)	**94.27**	**68.18**	**78.27**	**31.21**	86.28	**55.54**	**54.84**	**17.06**		**73.18**	**42.27**	**62.69**	**23.79**	**67.74**	**35.67**	**43.47**	**14.84**

Table 3: **Cross-dataset 3D assembly results**, with order prediction, free-space start for IKEA-Manual. $_{\text{sim}}$: AssemblyBench's physics-simulator results. The scale of SCD is 10^{-3}. *: fully-supervised, except for IKEA-Manuals. †: prompted zero-shot with the gt assembly order.

	AssemblyBench					IKEA-Manual		
Method	SCD↓	PA_{sim}(%)↑	SR_{sim}(%)↑	PA(%)↑	SR(%)↑	SCD↓	PA(%)↑	SR(%)↑
Vision-language models								
Qwen3-VL-8B†	–	–	–	15.12	1.43	14.97	5.83	0.00
Qwen3.8-27B†	–	–	–	22.18	6.43	17.63	3.84	0.00
Specialized models								
Manual-PA*	4.24	23.24	1.79	70.04	33.57	20.56	44.31	6.58
AssemblyDyno*	3.91	42.76	13.57	71.21	34.64	–	–	–
Ours								
AssemblyMind (SR-SFT)	3.00	49.37	18.21	73.83	44.29	**5.44**	73.93	22.37
AssemblyMind (GV-GRPO)	2.83	**49.48**	**19.24**	**74.19**	**45.35**	5.45	**75.77**	**23.68**

zero-shot transfer. Evaluation metrics and definitions follows Section 3.2. The part of each step is either provided with *ground-truth order* or *predicted order*.

Baseline Methods. We compare AssemblyMind with general vision-language models and specialized assembly models. For specialized models, AssemblyDyno (Li et al., 2026) is retrained on AssemblyVerse by predicting displacements instead of absolute poses, from reference starts only and with the same staging randomization augmentation (+SA) as ours. We retrain Manual-PA (Zhang et al., 2025) as well and interpolate linearly from the starting pose to the final pose. We also test on open-sourced VLMs, i.e., Qwen3-VL 2/8B, Qwen3.8 27B (zero-shot), for a comprehensive evaluation. See implementation details in Appendix. A.

5.1 Main Results

As seen in Table 2, With the given order, AssemblyMind achieves strong final-pose accuracy and generates valid trajectories for most parts, substantially outperforming AssemblyDyno trained on the same data. The gap is largest as the starting pose deviates from the reference configuration. All other baseline models struggle with accurate final-pose prediction and collision-aware trajectory generation. AssemblyMind also shows strong zero-shot generalization across assemblies of varying complexity, outperforming specialized models trained on the same dataset, as shown in table 3.

We present qualitative results and comparisons with AssemblyDyno in Fig. 6. We show single-step trajectories from free-space starts together with previous-step predictions, demonstrating accurate final-pose estimation and valid trajectory generation. We further show full assembly sequences and trajectories predicted by AssemblyMind in Fig. 5. **Qualitative video results are provided as part of the supplementary materials.**

5.2 Ablation Study

In Table 4, we present results of ablating every component of our model. 'Baseline' stands for the model trained with natural pose order, i.e., $[p_1,\ p_2, p_3, \ldots, p_K]$. From the table, we can see

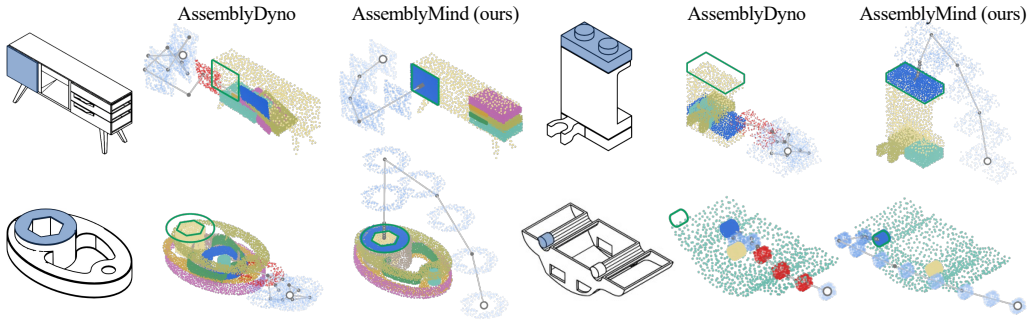


Figure 5: Qualitative results on the single-step assembly trajectory prediction. We compare with AssemblyDyno trained with AssemblyVerse on different assembly domains. Green boxes represent ground-truth final poses, and red point clouds represent invalid intermediate key poses (collisions).

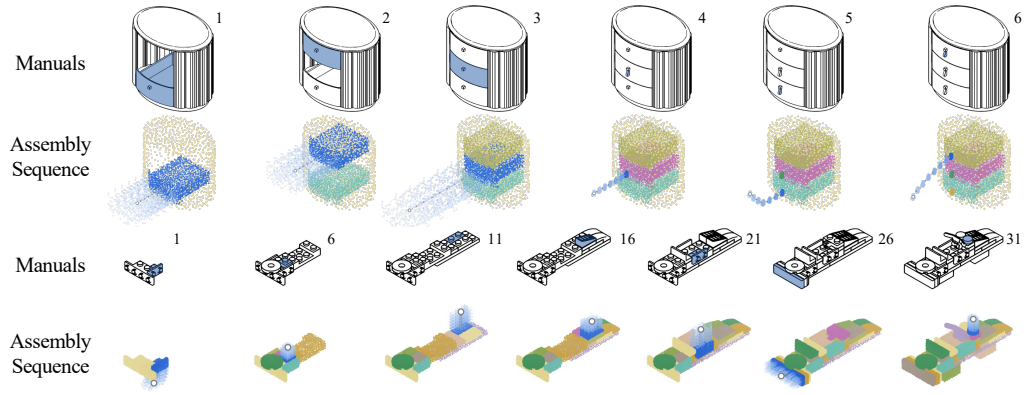


Figure 6: Qualitative results on full assembly sequence trajectory planning. The moving part in each step is shown using blue point clouds for visualization. The first example (*top*) shows free-space start, and the second (*bottom*) shows reference start. Some intermediate steps are skipped.

that a model trained from reference starting poses only is accurate only for those starting poses but degrades sharply from free-space starting poses. Staging randomization recovers most of the final-pose accuracy and more than doubles trajectory validity. GV-GRPO then improves trajectory validity further, again at comparable or slightly better final-pose accuracy. These results attest to the effectiveness of each component of our proposed AssemblyMind.

6 Conclusion

To summarize, we introduce **AssemblyVerse**, a large-scale, diverse, process-level, multimodal 3D assembly dataset for unified 3D assembly reasoning and trajectory planning where each step includes multimodal instructions, part geometry, a geometrically certified trajectory, and trajectory-level validity metrics. In addition, we propose **AssemblyMind**, a VLM-based method that autoregressively predicts the full trajectory of each part in the assembly. AssemblyMind combines pose-token trajectory generation, pose-token reordering, staging-randomized SFT, geometry-verified GRPO, and order grounding to enable robust state-conditioned assembly planning. These result in state-of-the-art performance across benchmarks, start conditions, and both given and predicted order, while also transferring zero-shot to unseen domains. Together, AssemblyVerse and AssemblyMind move multimodal assembly models beyond predicting where parts belong toward reasoning about—and executing—how complete objects are built, paving the way towards physics-aware 3D assembly in real-world embodied and robotic settings.

Table 4: **Ablations** study on different techniques proposed in our AssemblyMind framework.

	Reference			Free-space	
Model variant	PA	SR	TVR	PA	TVR
Baseline	71.9	33.1	54.9	12.9	3.1
+ pose-token reordering	93.5	**68.2**	73.5	44.4	20.2
+ staging randomization	94.2	67.8	75.4	86.3	49.8
+ GV-GRPO	**94.3**	**68.2**	**78.3**	86.3	**54.8**

AI USE STATEMENT

In this work, we used generative AI tools for drafting and editing portions of the manuscript, improving writing clarity and readability, and help with implementation debugging. We also used generative AI tools for generating part of the annotations in the dataset. We have not used generative AI tools for proving mathematical claims, and generative AI-assisted idea collections are not applicable to this work. We have reviewed all AI-assisted work, and the LLM generated code and annotations in the dataset were verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies multimodal instruction grounding and 3D trajectory generation for assembly tasks. Our experiments do not involve human assemblies or the collection of personal information. The construction and release of AssemblyVerse are subject to the applicable licenses and redistribution permissions of the source assets. While our geometric verification procedure evaluates trajectory feasibility and collision avoidance in simulated environments, it does not guarantee safe execution on physical robotic systems. Real-world deployment would require additional validation and appropriate safety measures.

REPRODUCIBILITY STATEMENT

We provide implementation and experimental details in the main paper and appendix to facilitate reproducibility. These include the construction and processing of AssemblyVerse, the geometric verification procedure, and the architecture of AssemblyMind. We further describe the training configurations for staging randomization SFT (SR-SFT) and geometry-verified GRPO (GV-GRPO), together with the evaluation protocols, metrics, and other experimental details. Additional implementation details and experimental analyses are provided in the appendix.

REFERENCES

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

Blender Online Community. Blender – a 3d modelling and rendering package. https://www.blender.org/, 2025.

Yuening Cai, Junwei Zhou, Youran Qu, and Yu-Wing Tai. Ideal-bench: Indoor dataset and evaluation suite for analyzing 3d layout reasoning. *arXiv preprint arXiv:2607.03614*, 2026.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, 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*, pp. 14455–14465, 2024.

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12200–12209, 2021.

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. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

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

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.

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

Zhi Jing, Jinbin Qiao, Ouyang Lu, Jicong Ao, Shuang Qiu, Huazhe Xu, Yu-Gang Jiang, and Chenjia Bai. Assemlm: A spatial reasoning multimodal large language model for robotic assembly. *arXiv preprint arXiv:2604.08983*, 2026.

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

Danrui Li, Jiahao Zhang, Bernhard Egger, Moitreya Chatterjee, Suhas Lohit, Tim K Marks, and Anoop Cherian. Assemblybench: Physics-aware assembly of complex industrial objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Yunong Liu, Cristobal Eyzaguirre, Manling Li, Shubh Khanna, Juan Carlos Niebles, Vineeth Ravi, Saumitra Mishra, Weiyu Liu, and Jiajun Wu. Ikea manuals at work: 4d grounding of assembly instructions on internet videos. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. *URL https://arxiv.org/abs/2503.20783*, 1, 2024b.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 909–918. IEEE, 2019.

Yu Qi, Yuanchen Ju, Tianming Wei, Chi Chu, Lawson LS Wong, and Huazhe Xu. Two by two: Learning multi-task pairwise objects assembly for generalizable robot manipulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17383–17393. IEEE, 2025.

Silvia Sellán, Yun-Chun Chen, Ziyi Wu, Animesh Garg, and Alec Jacobson. Breaking bad: A dataset for geometric fracture and reassembly. *Advances in Neural Information Processing Systems*, 35: 38885–38898, 2022.

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

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Zirui Song, Guangxian Ouyang, Mingzhe Li, Yuheng Ji, Chenxi Wang, Zixiang Xu, Zeyu Zhang, Xiaoqing Zhang, Qian Jiang, Fengxian Ji, et al. Maniplvm-r1: Reinforcement learning for reasoning in embodied manipulation with large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18558–18566, 2026.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl DD Willis, and Wojciech Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Transactions on Graphics (TOG)*, 41(6):1–11, 2022.

Yunsheng Tian, Karl DD Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, et al. Asap: Automated sequence planning for complex robotic assembly with physical feasibility. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4380–4386. IEEE, 2024.

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. *arXiv preprint arXiv:2502.10090*, 2025.

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, 2022a.

Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Ran Zhang, Chin-Yi Cheng, and Jiajun Wu. Ikea-manual: Seeing shape assembly step by step. *Advances in Neural Information Processing Systems*, 35:28428–28440, 2022b.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5): 1–12, 2019.

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

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11094–11104. IEEE, 2020.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. PointLLM: Empowering large language models to understand point clouds. In *European Conference on Computer Vision (ECCV)*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, Hao Dong, et al. Generative 3d part assembly via dynamic graph learning. *Advances in Neural Information Processing Systems*, 33:6315–6326, 2020.

Jiahao Zhang, Anoop Cherian, Cristian Rodriguez, Weijian Deng, and Stephen Gould. Manual-pa: Learning 3d part assembly from instruction diagrams. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6304–6314. IEEE, 2025.

Junwei Zhou and Yu-Wing Tai. Gena3d: Generative amodal 3d modeling by bridging 2d priors and 3d coherence. In *European Conference on Computer Vision (ECCV)*, 2026a.

Junwei Zhou and Yu-Wing Tai. Perceive-then-plan: Layout-as-policy for monocular 3d scene layout estimation. *arXiv preprint arXiv:2605.25326*, 2026b.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5738–5746. IEEE, 2019.

A More Implementation Details

In this section, we provide more implementation details for our paper, including the dataset generation pipeline, the detailed implementation of our AssemblyMind, and the training & evaluation proxies. We provide them as follows:

A.1 Dataset Generation.

Sources and conversion. Each source corpus is converted into per-part triangle meshes: URDF trees for ArtiVerse and PartNet-Mobility, brick meshes for BrickNet, resolved world transforms for the Fusion 360 Gallery, and tessellated STEP geometry for the inverters. Every assembly is then put in one frame: the parts are translated by the assembly's bounding-box centre and divided by its bounding-box diagonal, so that every assembly has diagonal 1 and is centred on the origin, and a per-domain rotation puts its canonical up-axis on $+z$. This is the frame in which the pose tokens, the chamfer threshold and the clearance tolerances of Section 3.2 are all defined.

Planning. The planner removes one part per round in three stages: propose, certify, choose. Candidates are ranked by geometric priors, excluding parts that a support map pins. Each candidate gets a proposed trajectory: a templated axial or helical extraction for fasteners, otherwise a straight line searched over the six global axes followed by 64 Fibonacci-sphere directions, ordered by clearance against the local contact normals. Certification marches the trajectory against every remaining part with an exact collision check, backed by a swept-point test that catches thin-feature tunnelling; interference already present in the assembled state, such as flush faces and press fits, is tolerated as long as it does not deepen, which is what allows real CAD assemblies to be planned at all. When no candidate yields a certified straight line, an RRT-Connect escape planner supplies the trajectory. A VLM then chooses among the *certified* candidates only, and an assembly is released only if *every* step was accepted, rather than padded out.

Key poses and staging. The certified removal path is reversed to become the assembly trajectory and resampled to $K{=}12$ key poses: both endpoints, then the interior vertices with the highest corner score, then arc-length fill, with rotations interpolated by SLERP and the shortest segments merged first. The seated pose is forced to be the exact last vertex of the trajectory, never an interpolated one, and is re-checked against the part's stored pose. The first key pose is the staging pose the task conditions on; it is the pose the part reaches at the end of its certified extraction, so the reference start is guaranteed to have a collision-free straight route to the seat. The templated unscrewing trajectories record an axis and a travel distance but no turn angle, so the $8{,}465$ steps a human would unscrew ship as pure translations along the thread axis.

Rendering. Every assembly is rendered step by step at $768{\times}768$ in six orientations under one fixed orthographic camera. Each step and view produces the instruction diagram, in one of three line-art styles chosen per domain, together with a flat per-part colour pass, a motion-arrow pass and the per-part poses. The colour pass is what locates a part in a view: a part's mask is cut by matching its assigned colour, so every mask is exact rather than estimated. The difference image between two consecutive diagrams is the same object the model consumes as D_t, and the number of pixels it changes is what marks a step as blank or marginal in that view. A view-selection stage records, for every step, the upright view that shows the most of the part being added, penalising views in which the part touches the image border.

Names and instructions. Look-alike parts are grouped by their bounding-box extents before naming, so that a group receives one name. ArtiVerse takes its names from the URDF link names and BrickNet from the LEGO catalogue; the remaining three domains are named by a vision-language model: GPT-5-mini for Fusion 360, Qwen3-VL-8B-Instruct for PartNet-Mobility and the inverters, which sees the part highlighted in its step, in the finished assembly and beside the other members of its group, and is forbidden colours, viewer-relative words and names already used in the assembly. The instructions are written by Qwen3.5-9B, one sentence per step, from the diagram with the moving part highlighted, named and carrying its insertion arrow, a close-up of that part, and a reference frame in which every part placed so far is labelled; the prompt forbids viewer- and camera-relative language and asks for contact-based relations instead. When no view shows any pixel of the moving

part, the sentence is written without an image, from the part's name and its mesh-verified mating partners, and the prompt forbids any invented location, orientation or mechanism. The first step of every assembly is a fixed sentence rather than a generated one.

A.2 OUR MODEL: ASSEMBLYMIND.

The backbone is Qwen3-VL-2B-Instruct, extended with the point-cloud encoders below and then fine-tuned in full: all 2.14 B parameters are updated, the vision tower included, and no low-rank adapter is used in the SFT cold start stage.

A trajectory is $K{=}12$ key poses of $D{=}9$ dimensions, rotation before translation within a pose, quantised into $B{=}256$ bins over $[-1, 1]$ (bin width $2/255$) in the render frame, in which the assembly's bounding-box diagonal is 1. Each bin is a dedicated token <assemble_pose_b> added to the vocabulary together with the three point-cloud markers, so a trajectory is 108 pose tokens between <traj_start> and <traj_end>. The seat-first serialisation is the permutation $[1, K, 2, \ldots, K{-}1]$ of the key poses; the first nine tokens are the observed start pose, supplied as context, masked from the loss and force-fed at inference, so 99 tokens are predicted. The nine tokens of the final pose carry $w_{final}{=}5$, to make the final pose estimation results precise enough to guide the remaining trajectory key poses generation. Each step contributes two $512{\times}512$ images, the diagram before the step and the difference image, which the vision tower turns into 256 tokens each; the dataset's 768 px renders are resized once, not cropped.

The moving part and the union of the already-placed parts are each sampled to $1{,}024$ points (area-weighted surface samples reduced by farthest-point sampling) and encoded by a shared VN-DGCNN with a 20-nearest-neighbour graph. Its equivariant output is read as $L{=}256$ tokens of 12 dimensions and projected into the language model's embedding space by a three-layer MLP, giving 512 point tokens per step. Encoder and projector are trained with the rest of the model, at their own learning rates (i.e., 1e-5).

The order-grounding head scores every (step, part) pair of an assembly from frozen precomputed features. On the step side, four streams—the instruction embedding, the mean diff-image and after-image embeddings, and a 16-cell diff grid pooled by a learned attention query—are each projected by a LayerNorm–Linear–GELU–Dropout–LayerNorm block to width $d = 512$ and combined by a softmax gate computed from their concatenation; a learned step-position embedding is added and the sequence of step queries q_t is contextualised by a two-layer pre-norm transformer encoder (8 heads, FFN width $4d$) over the whole manual. On the part side, three geometry streams—rotation-invariant shape features, mesh-equivalence features, and backbone token features—are projected to d, summed, and contextualised by a two-layer set transformer over the assembly's parts, giving h_i. A candidate is scored by an MLP on $[\, q_t,\ h_i,\ q_t \odot h_i,\ |q_t - h_i|\,]$, yielding an $N \times P$ score matrix. Dropout is 0.1 and the head has 23.4 M trainable parameters.

A.3 TRAINING & EVALUATION.

SFT cold start. We use AdamW with no weight decay, a cosine schedule with a 3% warmup, and four learning rates: 5×10^{-5} for the language model and the vision tower, 1×10^{-5} for the point encoder, 1×10^{-4} for its projector, and 1×10^{-3} for the new embedding rows, whose gradients are masked so that the rest of the embedding matrix is untouched. We run the training with 6 epochs and a batch size of 16, a visibility-based filtering is applied to the training set to make the visual inputs are effective.

GV-GRPO stage. We apply the LoRA with rank-64 adapter ($\alpha{=}128$, 25.7 M parameters) on the attention projections of the language model; with the adapter disabled the model is bit-exact the supervised one, which is therefore the reference policy for the KL term. We sample $G{=}8$ rollouts per prompt at temperature 0.8, 32 prompts per optimiser step, and resample only the translation dimensions of the pose tokens, holding the rotations and the forced start prefix. The objective uses $\epsilon{=}0.2$, $\beta{=}0.02$ and mean-centred advantages without standard-deviation normalisation; the learning rate is 5×10^{-6} on a cosine schedule.

Order-grounding head. Trained on frozen features for 12 epochs with AdamW, batch 16 assemblies, learning rate 3×10^{-4}, weight decay 0.05 and a 5% warmup.

Path validity (TVR). The predicted polyline is probed every 0.03 diagonals along each straight segment between consecutive key poses, with the final contact at the seat excluded, using 256 of the part's points against a kd-tree of the placed parts. Each probe is compared with the probes of the certified reference path that lie within 0.02 of the same distance to the seat: where that reference is in free space the probe must clear the scene by τ=0.005, and where the reference is itself in contact it must clear it by at least the reference's own clearance minus δ=0.004, half a translation bin, which is the finest placement the tokeniser can express. A path is valid only if no probe fails.

Start conditions. On the training side, we additionally add a perturbation to the planner start, together with the reference start and starting randomization, at a mixture rate of 30%, 30%, 40%. The perturbed start adds Gaussian noise of σ=0.15 assembly diagonals to its translation and accepts a draw only if both the displaced part and the straight segment back to the planner's start clear the placed parts by 0.005; after eight rejected draws the clean start is kept, so the condition never introduces a start already in collision. On the evaluation side, the reference start is the planner's own staging pose. A free-space start moves the translation onto a routing shell of radius $1.2\max(R_c,\ \|k_0\| + r + \varepsilon)$, where R_c is the collision envelope of the placed parts, r bounds the moving part and ε=0.025, along one of the six assembly-frame axes.

Look-alike parts. Two thirds of all parts sit in a group of interchangeable instances, so the metrics treat a group as one part. Parts are grouped by mesh equivalence: matching vertex and face counts, a rotation-invariant shape descriptor, matching chirality, so that mirror pairs are not merged, and matching histograms of pairwise vertex distances. Grounding is supervised and scored on groups rather than part identities, and the order number we report as KD is the group-aware Kendall τ_a after instances within a group have been matched to the reference assignment on AssemblyVerse, so higher is better. Baselines that predict a pose per part without part identity are matched to the ground truth inside each group before scoring, so the grouping never penalizes them.

Retraining AssemblyDyno. AssemblyDyno predicts a final pose and a trajectory but takes no start pose, so it cannot be evaluated on our task as published (or final pose-only, see table 3). We change it in the smallest way that makes it comparable: its trajectory head predicts each key pose as a displacement from key pose 0 and the given start pose is composed back in ($t_k = t_0 + \Delta t_k$, $q_k = \Delta q_k\, q_0$), with all losses and metrics on the composed poses. On top of that we train two variants. The *reference-start* variant additionally embeds the start pose: the same nine-vector AssemblyMind receives, through a zero-initialised MLP added to each part token, and is trained on planner starts only. The *starting randomized* variant is the same model trained on exactly our start mixture: 35% free-space starts and otherwise a clearance-checked jitter of key pose 0, with validation and test left unaugmented. Both trained with six epochs on AssemblyVerse, the budget AssemblyMind was trained under, and the same training recipe on AssemblyBench.

Comparison methods. For the Vision-Language Model baselines compared in our experiments, we perform zero-shot inference on these models. Since we common VLMs do not have the ability to access the point cloud information, we do not input point clouds into the models. Each step is presented as two 512 px renders—the assembly before the step, and the target configuration after it. Note that we pass the plain after-render rather than the before/after pixel difference used by our own image encoder: at 512 px that difference is almost entirely black (0.0–0.1% informative pixels), so a general-purpose VLM would receive no goal information from it. The model is asked for the part's *final seated pose* alone, as strict JSON ({`"t": [...], "q": [...]`}), and we decode greedily. As the same setting of our AssemblyMind, we also perform autoregressive prediction with the VLM baselines, which generate each step's final poses and use the generated final pose, place the corresponding part for the next step's input.

B More dataset examples

In this section, we provide more dataset examples, and provide a detailed breakdown of the dataset by different domains, and different difficulties.

Composition. Table 5 breaks the corpus down by source domain. The four public domains contribute a similar number of assemblies each, and the industrial inverters add a smaller fifth domain, but their part statistics differ sharply: PartNet-Mobility assemblies have a median of three parts and ArtiVerse five, whereas a LEGO set has a median of eighteen, a Fusion 360 assembly fifteen and

Table 5: **AssemblyVerse by source domain.** Parts per assembly and the assembly-level splits. "Parts" counts every part of every assembly: the base part, which is given, and one moving step for each of the others.

	Size		Parts / assembly			Assemblies per split			
Domain	Assemblies	Parts	median	mean	max	train	val	test	excl.
ArtiVerse (articulated objects)	2,816	24,513	5	8.7	118	2,367	139	279	31
BrickNet (LEGO sets)	2,740	74,663	18	27.2	100	2,323	137	273	7
Fusion 360 Gallery	2,607	70,043	15	26.9	540	2,158	127	254	68
PartNet-Mobility	2,287	13,819	3	6.0	116	1,839	108	216	124
Inverter (industrial)	359	9,374	18	26.1	212	283	17	33	26
AssemblyVerse	10,809	192,412	8	17.8	540	8,970	528	1,055	256

Table 6: **Difficulty tiers.** Assemblies per parts-per-assembly tier, by domain.

Parts	ArtiVerse	BrickNet	Fusion 360	PartNet	Inverter	All
2–4	942	102	196	1,757	37	3,034
5–10	1,239	712	754	313	56	3,074
11–20	506	651	682	134	154	2,127
21–40	80	641	509	33	56	1,319
41–80	19	499	314	21	39	892
81+	30	135	152	29	17	363
All	2,816	2,740	2,607	2,287	359	10,809

an inverter eighteen, with a long tail up to 540 parts. Splits are drawn per assembly and stratified by domain, in the ratio $0.85/0.05/0.10$; assemblies that fail the release gate or have no manual are assigned to no split.

C More Results

C.1 More quantitative results.

AssemblyVerse pools five corpora with very different part statistics, and because SR and E2E are all-or-nothing per assembly, we give the two breakdowns that the pooled number hides: by source domain (Table 7) and by the number of parts in the assembly (Table 8).

The domain ordering is the same under every start condition and for both models. Fusion 360 is the hardest of the five domains, together with the inverters, they are the only domains on which PA drops below 90 from the reference start: their assemblies are large, their parts are often small fasteners or complex parts, which makes the final pose estimation task much more complex compared to other domains. PartNet-Mobility is the easiest, but its assemblies have a median of three parts, so its SR and E2E are not comparable with BrickNet's or Fusion 360's. GV-GRPO leaves PA and SR comparable or slightly better in all domains and raises TVR in all of them, by $+2$ to $+5$ points, which has a obvious improvement on the trajectory validity.

Table 8 separates the two things the pooled SR and E2E confound. PA falls gently with assembly size, from 98.1 on assemblies of at most four moving parts to 86.6 on assemblies of more than fifty, a per-part accuracy that degrades slowly, which exactly reflects the relationship between the difficulty and assembly sequence lengths. TVR is almost flat across the size bins, so a longer assembly does not make an individual insertion harder to reason about. SR and E2E, in contrast, collapse, since the auto-regressive prediction paradigm they will accumulate per-part erros, which makes it much more harder to get a successful and complete assembly prediction: at 94% per part, an assembly of fifty parts succeeds completely only about 5% of the time even if the parts were independent.

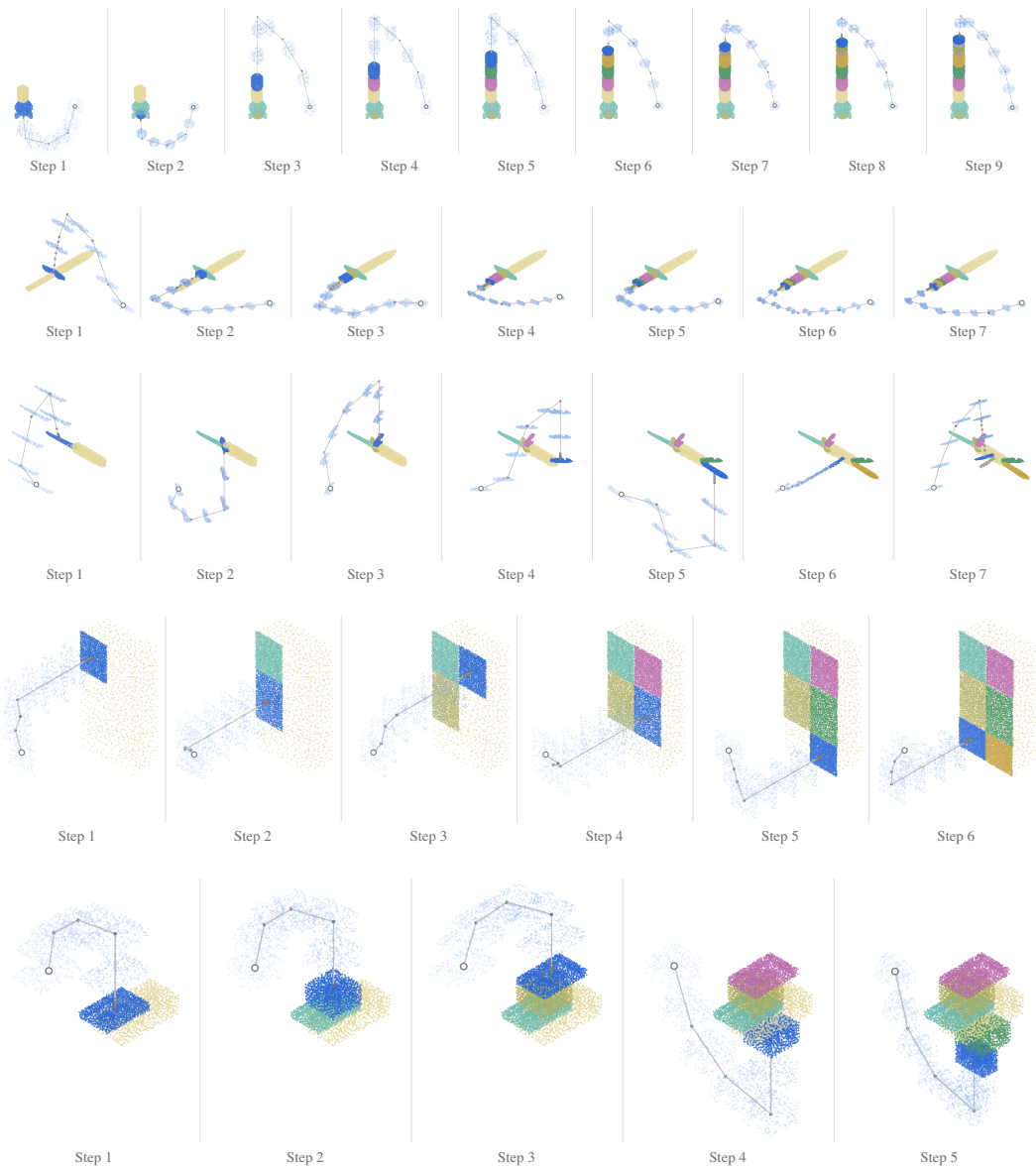


Figure 7: We present more qualitative results on the whole assembly planning results with free-space start assembly trajectories, including different domains.

C.2 More qualitative results.

Figures 7 show complete assembly prediction results. Every panel is one step of a auto-regressive rollout: the parts placed in earlier steps sit where the model itself put them, not at their ground-truth seats, so an error made early is visible in every later panel. The trajectory of the part added in a step is drawn as a sequence of ghosts along its predicted key poses.

C.3 Dataset quality check.

The instruction text of AssemblyVerse is written by a model, so we audit it with a second, stronger model that did not take part in the generation. We sample 5000 steps from the released corpus, allocated across the five domains in proportion to each domain's share of the 182,909 released steps and drawn uniformly at random within each domain, and present each one to Claude Fable 5.1 as the rendered state before and after the step together with the sentence the dataset ships for it. The

Table 7: **AssemblyVerse test by source domain**, given order, chained. *Free-space* is the mean over the six canonical shell starts.

			AssemblyMind				AssemblyMind + GV-GRPO			
Domain	Asm.	Parts	PA	SR	TVR	E2E	PA	SR	TVR	E2E
Reference start										
ArtiVerse	279	2,469	97.5	86.0	74.4	30.8	97.5	86.7	77.9	35.5
BrickNet	273	6,894	96.8	66.3	78.3	13.2	96.7	66.3	81.0	15.4
Fusion 360	254	6,627	85.3	33.1	69.1	9.4	85.4	33.2	71.0	11.0
PartNet-Mobility	216	1,144	98.0	92.1	80.6	63.9	98.1	92.6	85.3	69.4
Inverter	33	897	88.7	33.3	73.1	12.1	84.7	27.4	71.1	15.2
All	1,055	18,031	94.2	67.8	75.4	27.3	94.3	68.2	78.3	31.2
Free-space start, mean of six axes										
ArtiVerse	279	2,469	95.3	80.6	57.0	17.6	95.4	80.9	61.7	21.3
BrickNet	273	6,894	88.8	41.7	44.1	1.3	88.2	39.9	50.1	2.1
Fusion 360	254	6,627	69.4	18.7	36.6	1.0	69.5	18.5	39.5	1.3
PartNet-Mobility	216	1,144	95.0	86.2	65.1	46.4	94.8	86.4	70.0	51.8
Inverter	33	897	66.2	15.7	36.8	0.0	66.4	19.7	41.5	0.0
All	1,055	18,031	86.3	54.7	49.8	14.8	86.3	55.5	54.8	17.1

Table 8: **AssemblyVerse test by assembly sequence length (difficulty)**, given order, chained, on the full test split (1,055 assemblies, all five domains). Bins are the number of moving parts, i.e. the number of predicted steps.

			AssemblyMind				AssemblyMind + GV-GRPO			
Moving parts	Asm.	Parts	PA	SR	TVR	E2E	PA	SR	TVR	E2E
Reference start										
2–4	374	920	98.1	94.1	78.2	57.0	98.2	94.7	82.6	64.2
5–10	240	1,725	94.0	74.6	73.6	21.7	94.0	75.0	76.2	23.8
11–20	187	2,761	93.8	58.3	74.6	9.1	93.2	58.3	76.5	10.2
21–50	168	5,544	90.3	34.5	73.6	1.8	89.9	35.1	75.3	3.6
51+	86	7,081	86.6	19.8	73.5	3.5	86.3	16.3	75.8	2.3
Free-space start, mean of six axes										
2–4	374	920	95.4	88.1	61.6	39.2	95.5	88.3	66.6	44.3
5–10	240	1,725	87.0	58.2	46.6	3.1	87.5	59.3	51.9	4.9
11–20	187	2,761	83.3	39.5	43.0	0.7	82.8	36.4	47.0	1.3
21–50	168	5,544	77.3	15.3	40.5	0.3	76.4	14.2	44.4	0.2
51+	86	7,081	70.0	10.5	40.6	0.0	69.7	10.5	44.4	0.0

judge is asked to rate the sentence on three axes independently: whether it identifies the right part (*grounding*), whether the spatial relation it asserts holds in the geometry and 3D-aware information from the rendered manuals (*correctness*), and whether it is free of viewer-relative and colour-based references (*style*). Each axis is scored on a 1–5 scale, and we report the distribution rather than the mean alone, since a small number of badly wrong sentences matters more than a shift in the average.

Style is the strongest axis and grounding the next, with 85.4% of sentences at 4 or 5 and none at 1; grounding failures concentrate on steps whose part is largely occluded in the diagram the sentence was written from. Correctness is bounded above rather than wrong: 87.5% of steps assert nothing the geometry contradicts and only 0.9% are plainly false, while 37.9% score 3 because the sentence is generic enough to be unfalsifiable at render scale.

We provide the prompt input to the judge model as follows:

Table 9: **Instruction quality audit.** Claude Fable 5.1, 5000 sampled steps from the released corpus, stratified by domain.

	Score distribution (%)					
Axis	1	2	3	4	5	Mean
Grounding	0.0	2.5	12.1	39.2	46.2	4.29
Correctness	0.9	5.6	31.9	33.2	28.4	3.83
Style	0.0	2.9	0.4	13.3	83.3	4.77

Instruction-quality judging prompt

```
You are an expert evaluator of assembly instruction manuals. You will be
given two renders of one assembly step, the state before the step and the
state after it, together with the one-sentence instruction the dataset ships
for that step.

Your task is to rate the sentence on three axes, independently of one
another. Judge only from what the images show. Do not reward or penalise
fluency, grammar or length.

1. Grounding: does the sentence refer to the part that this step actually
adds?
  5: unambiguous referent
  4: correct referent under a generic or imprecise name
  3: ambiguous, or the referent cannot be confirmed from the images
  2: mostly refers to some other part
  1: names a different part entirely

2. Correctness: does the spatial relation the sentence asserts hold in the
geometry and 3D information shown in the rendered manuals?
  5: every asserted relation is verified
  4: the main relation holds, a minor qualifier is vague
  3: generic or near-vacuous, or only partly verifiable
  2: an asserted relation is doubtful or partly contradicted
  1: plainly false
  NA: the step changes no pixels, so the images carry no evidence

3. Style: is the sentence free of viewer-relative and colour-based
reference?
  5: none
  4: a directional word anchored to a named part with a canonical face
  3: a directional word with no clear anchor
  2: one clear violation: an explicit viewer- or camera-relative phrase, or a
     colour word used to identify a part
  1: two or more violations

Return one score per axis and one line of justification for each.
```

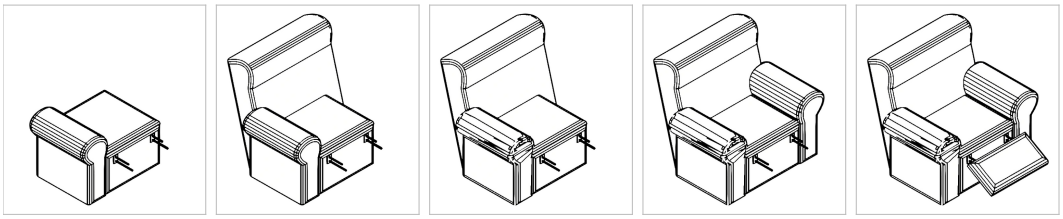
D LIMITATION & FUTURE WORKS

Our work focuses on assembly with rigid parts and assumes that the relevant parts, manuals, and observations are available at each step. While this setting captures a broad range of structured assembly tasks, it does not yet cover deformable objects, articulated components with complex kinematic constraints, or cases requiring additional sensing and active viewpoint selection. Moreover, our geometric verifier evaluates trajectory feasibility and collision avoidance in simulation, but does not model all sources of uncertainty encountered by physical robots, such as perception noise, calibration error, grasp uncertainty, contact dynamics, and execution drift.

AssemblyMind also performs trajectory prediction sequentially, so errors made at earlier steps may propagate through the remaining assembly process. The current formulation does not explicitly support recovery, backtracking, or long-horizon replanning when intermediate states deviate from the expected configuration. Future work could therefore extend the framework toward closed-loop physical execution, incorporating online state estimation, uncertainty-aware planning, failure detection, and corrective actions. Another promising direction is to move beyond step-wise assembly toward more general embodied spatial reasoning, where an agent jointly interprets multimodal in-

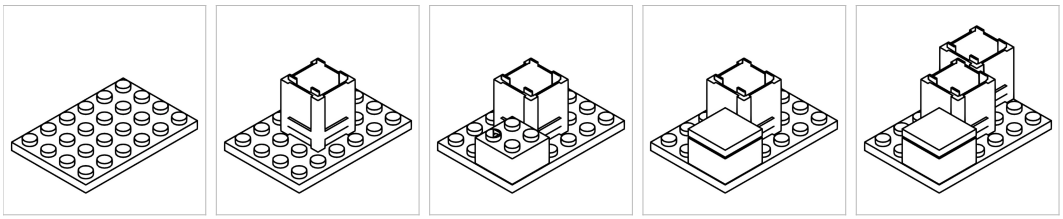
structions, reasons about object interactions and physical constraints, and plans long-horizon actions in dynamic 3D environments.

ArtiVerse armchair



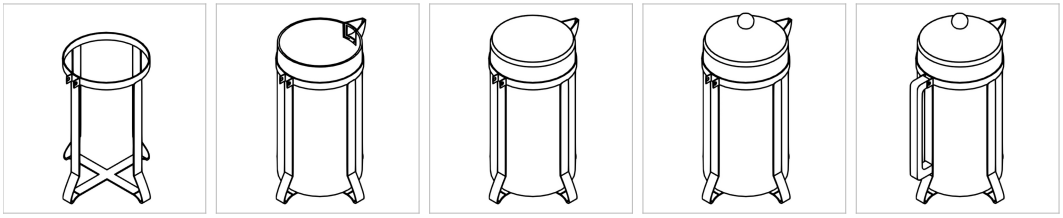
Place Base in the center.	Lower the Backrest until it rests against the Base.	Attach the Armrest to the side of the Backrest.	Attach the Armrest to the side of the Base.	Slide the Footrest into the slot located at the front of the Base.

BrickNet (LEGO) crate build



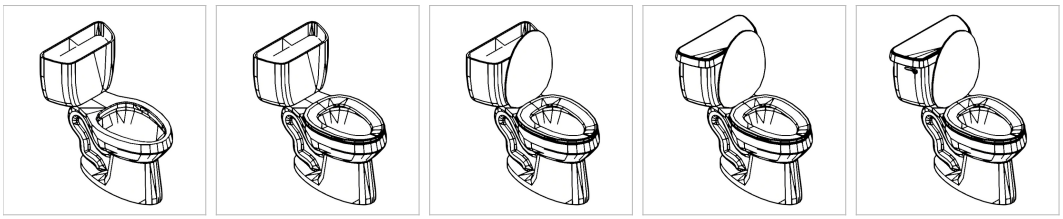
Place plate 4x6 in the center.	Place the container 2x2x2 crate into the plate 4x6.	Place the brick 2x2 on top of the plate 4x6.	Place the tile 2x2 with groove on top of the brick 2x2.	Place the container 2x2x2 crate into the base structure.

Fusion 360 canister press



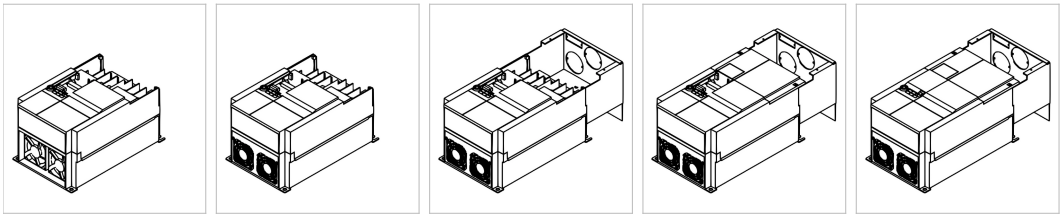
Place Support Frame in the center.	Position the Canister into the assembly.	Place the Lid onto the top of the Canister.	Press the Plunger Knob down into the Lid.	Attach the Handle to the side of the Canister.

PartNet-Mobility toilet



Place Tank in the center.	Place the Seat onto the bowl rim.	Lower the Lid onto the Seat.	Lower the Cistern Cover onto the top of the Tank.	Attach the Flush Handle to the Cistern Cover.

Inverter power inverter



Insert the Fan into the designated slot on the side of the Base Plate.	Align the Fan Cover with the opening in the Base Plate and slide it into place.	Insert the Starter Relay into the designated slot on the Base Plate.	Position the Cover Plate over the exposed internal components and lower it into place.	Place the Cover onto the top surface of the Base Plate.

Figure 8: We present more per-domain examples in our AssemblyVerse dataset, including the per-step rendered manuals and language instructions.