

AssemblyBench: Physics-Aware Assembly of Complex Industrial Objects

Danrui Li^{1*} Jiahao Zhang^{2*} Bernhard Egger³
Moitreyea Chatterjee⁴ Suhas Lohit⁴ Tim K. Marks⁴ Anoop Cherian⁴

¹Rutgers, The State University of New Jersey, USA ²The Australian National University, Australia

³Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany ⁴Mitsubishi Electric Research Laboratories (MERL), USA

¹danrui.li@rutgers.edu ²jiahao.zhang@anu.edu.au ³bernhard.egger@fau.de ⁴{chatterjee, slohit, tmarks, cherian}@merl.com

<https://merl.com/research/highlights/assemblybench>

Abstract

Assembling objects from parts requires understanding multimodal instructions, linking them to 3D components, and predicting physically plausible 6-DoF motions for each assembly step. Existing datasets focus on simplified scenarios, overlooking shape complexities and assembly trajectories in industrial assemblies. We introduce AssemblyBench, a synthetic dataset of 2,789 industrial objects with multimodal instruction manuals, corresponding 3D part models, and part assembly trajectories. We also propose a transformer-based model, AssemblyDyno, which uses the instructional manual and the 3D shape of each part to jointly predict assembly order and part assembly trajectories. AssemblyDyno outperforms prior works in both assembly pose estimation and trajectory feasibility, where the latter is evaluated by our physics-based simulations.

1. Introduction

Assembling objects from constituent parts is a challenging yet ubiquitous task with substantial potential for automation, and it has myriad applications from household furniture assembly to large-scale manufacturing of complex industrial objects. As a result of the advancements in large vision-and-language models and robotics foundation models, the problem of object assembly has garnered significant interest recently in both the computer vision and robotics communities [16, 30, 33, 35, 43]. State-of-the-art approaches to address this task mainly consider IKEA-style furniture assembly, due to the availability of abundant collections of well-designed instruction manuals that detail each step of the process using language-free diagrams [35, 38, 43]. Furthermore, to facilitate assembly by inexperienced users, furniture parts are typically designed to be easily distinguishable, clearly illustrated in diagrams,

*Work done during internships at MERL.

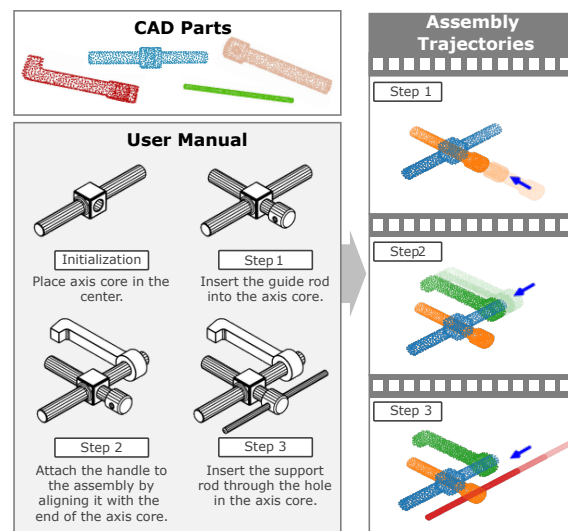


Figure 1. Given a step-wise manual with diagrams and text (lower left), we aim to assemble the corresponding set of 3D parts (upper left) in a virtual environment, outputting its step-wise assembly trajectories, which can be rendered into 4D animations (right).

and straightforward to attach to each other. Thus, datasets derived from such furniture assemblies offer a simplified yet useful setup to study this complex reasoning task.

However, furniture assemblies alone may not capture the full spectrum of complexities in real-world assembly processes, especially the process of moving assembly parts. For example, the assembly of electrical appliances (e.g., air conditioners, ceiling fans, laundry machines), industrial equipment (e.g., motors, gear boxes, hydraulic pumps), or even simple interactive toys. These objects often contain parts with complex geometries, and they may require sophisticated maneuvers such as insertion with twisting to assemble. While there have been several attempts at capturing varied aspects of this complex problem—e.g., assembly from diagram-based instruction manuals [34, 35, 43], planning for robotic assembly [30, 33], and learning from video

demonstrations [1, 19, 42]—there is a need for datasets that capture more of the common challenges in assembly.

Toward this end, we present AssemblyBench, a novel synthetic assembly dataset. It consists of nearly 3K assemblies spanning several categories of objects (not merely furniture) and featuring industrial objects. In contrast to prior non-IKEA datasets [32, 43], AssemblyBench extends the dataset modalities from assembled shapes to a set of CAD parts, step-by-step instruction diagrams with text descriptions, and assembly motion trajectories. Moreover, we introduce a pipeline that can be generalized for automatic instruction manual creation for any type of industrial objects from their CAD assemblies, which are commonly provided in mechanical design specifications.

To address the challenges in AssemblyBench, we present *AssemblyDyno*, a novel transformer-based architecture that predicts the assembly order of the parts as well as their 6-DoF motion trajectories. Specifically, AssemblyDyno is trained to learn a soft attention between the order of the instructions and the 3D part point clouds (encoded using a point cloud encoder) to regressing a discrete sequence of $SE(3)$ transformations for each part, along which the part should move in order to successfully complete its assembly. The entire set of motion sequences is jointly predicted by AssemblyDyno in a single forward pass. Our model is trained in a supervised setting, using ground-truth sequences of part trajectories, by minimizing chamfer-distance-based losses that account for invariance to symmetries in the parts’ geometries.

In addition to utilizing the standard metrics for evaluating the performance of prior works on AssemblyBench (e.g., symmetric chamfer distance and final success rate), we present a novel evaluation that executes the predicted part motion trajectories in a physics simulator [22] to verify their physical feasibility. Our key insights are three fold: i) while our training scheme does not include the simulator-in-the-loop, our supervised training might implicitly capture the physical constraints for assembly; ii) although instruction manuals are usually created to follow physically plausible part assembly, there may be other, novel motion pathways that could lead to a correct assembly; and iii) there may be inaccurate or physically infeasible steps in the predicted assembly that cannot be identified unless executed under physical constraints. For example, a predicted motion may cause the part to get stuck in an intermediate position, which would prevent the remainder of the assembly from proceeding. While prior protocols measure success using only the point-cloud alignment of the predicted final assembly, our physics-simulator-based evaluation offers a complete verification of the part assembly order, motion trajectories, and physical realizability, bringing successful assemblies significantly closer to real-world enactment.

We present extensive experiments demonstrating vari-

ous aspects of AssemblyBench using AssemblyDyno. We find that a state-of-the-art baseline [43], which demonstrates nearly 60% success rates on furniture datasets, does not perform nearly as well (nearly 30% worse) on our challenging new dataset. In contrast, AssemblyDyno, with its incorporation of both diagrams and text descriptions from the instruction manual, leads to 12% improvements in the success rate of final pose estimate. Furthermore, AssemblyDyno predicts the assembly trajectories with better physics feasibility. It achieves about 33% success rate in a physical simulator by referencing to diagrams and texts with a trajectory smoothing loss, while the baseline method achieves only around 3%.

In summary, our main contributions include:

- **Dataset:** AssemblyBench that includes complex industrial part assemblies with multi-modal user manuals and assembly trajectories, produced using a VLM-based generative pipeline.
- **Model:** AssemblyDyno, a generalized feed-forward model that takes in multi-modal assembly steps and 3D part point clouds, predicting the parts’ assembly order, final poses, and 6 DoF assembly motion trajectories.
- **Evaluation:** A physics engine based protocol to evaluate the physical feasibility of predicted assembly trajectories, where AssemblyDyno shows state-of-the-art results.

2. Related Work

Assembly Datasets. A comprehensive assembly dataset should include diverse geometries and contact types, incorporate realistic physical interactions, be realizable in a physics simulator, and capture the full assembly process rather than only final poses. Existing shape datasets such as PartNet [21] and IKEA-based assembly datasets [1, 19, 34, 35, 38, 42, 43] have been widely used, with IKEA manuals providing canonical step-by-step diagrammatic supervision. However, these datasets include a limited set of furniture objects with similar part geometries and lack kinematic constraints. Broader datasets covering toys [27] or electronics [29], as well as datasets with real-world video annotations [8, 27, 29, 45], broaden category coverage but still focus mainly on high-level goals or final part poses.

Motivated by these limitations and following [32], we build on the Assemble-Them-All (ATA) dataset [31, 32], which contains nearly 5K industrial CAD models with explicit part-insertion relations and physics-based disassembly trajectories [31]. Prior work has used ATA [32, 46], but it lacks step-by-step manuals, standardized part/trajectory representations, curated splits, and evaluation protocols.

Recent efforts aim to reduce manual annotation cost via automatic manual-generation pipelines, including parametric systems [24, 25, 37] and VLM-based dataset enrichment [11, 15, 41]. CheckManual [20] further explores VLM-driven operation-manual generation. How-

ever, assembly involves multi-part interactions, occlusions, and nontrivial 3D insertions, making manual generation substantially more challenging. Consequently, most datasets [5, 38, 43] provide only final part poses, overlooking the trajectories required for complex assemblies. Our dataset and pipeline address these gaps by producing standardized representations, full assembly trajectories, and step-by-step manuals. See Table 1 for detailed comparison.

Assembly Step Prediction. There are numerous works that attempt to predict assemblies without manuals [17, 18, 26, 40]. However, given the joint discrete-and-continuous search space for finding the part to assemble and generating its assembly trajectory, it is usually difficult for such methods to generalize. There are also several recent works that use guidance from: i) final object renderings [16, 39, 46], ii) from step-wise manuals [34, 35, 43], or iii) assembly videos [19, 42], but mostly for IKEA-type furniture. Classical motion planning methods (e.g., RRT [14] and PRM [9]) have been explored for assembly via generating the motion trajectory from a predicted final part pose [46], including incorporating physical constraints [31, 32]. However, they are computationally expensive and require precise characterization of the physical constraints in the environment [12]. Thus, while we use a physics-based planner [31, 32] when generating our dataset, our model is trained in a supervised manner on the trajectories, implicitly learning the physics. Further, our use of a single forward pass to predict all assembly steps at once in discrete time steps is computationally efficient and robust.

3. Proposed Method

In its generalized form, an assembly task involves understanding the procedure from instruction manuals, identifying the object parts to be assembled at each step, and executing the assembly steps as depicted in the manual to fit the parts together following physically feasible motion paths. Following this recipe, we formulate our task as follows.

We are given an unordered set of N assembly parts as 3D point clouds $\{P_i\}_{i=1}^N$, with each part assumed to contain the same number of points), and a manual consisting of a sequence of assembly instructions denoted $(\mathcal{I}_1, \dots, \mathcal{I}_N)$, where each step involves adding one part. Our objective is to have a model that: i) predicts the assembly order by grounding the 3D parts to their instructions, i.e., producing part indices $(\hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_N)$ such that the part $\mathcal{P}_{\hat{\pi}_i}$ is associated with instruction \mathcal{I}_i , and ii) predicting the part motion trajectories for each assembly step as $((\hat{R}_i^k, \hat{t}_i^k) \in \text{SE}(3))$, where $i \in \{1, \dots, N\}$ represents the assembly step number, and $k \in \{1, \dots, T\}$ represents the time step within a part’s trajectory. The number of parts N is assumed to vary across objects, however the number of time steps T in each part’s assembly trajectory is considered fixed. Each instruction step \mathcal{I}_i in the manual consists of a diagram illus-

trating the assembly step and a free-form text description detailing the step. We use 2D line-drawing diagrams similar to IKEA manuals, showcasing 3D parts without textures in a fixed parallel projection. The assembly trajectory in assembly step i is represented as a sequence of T 6-DoF poses, $(\hat{R}_i^k, \hat{t}_i^k)$, which respectively represent the 3×3 rotation matrix and 3×1 translation vector of the part at the k th time-step of its trajectory. In our experiments, we use $T = 12$ for the number of time-steps.

This work requires us to implement three parts: i) building the AssemblyBench dataset using an automatic annotation pipeline, which adheres to the assembly process described above while incorporating complex assemblies (detailed in § 3.2); ii) proposing a novel transformer-based reasoning model, AssemblyDyno, which predicts an entire 3D assembly process from an instruction manuals (described in § 3.3); and iii) proposing a set of evaluation metrics that evaluates the entirety of the assembly performance (explained in § 3.4).





3.1. AssemblyBench Dataset

























AssemblyBench contains multimodal instruction manuals for 2789 assemblies in total, covering a wide range of categories including furniture, appliances, and mechanical components. Each manual provides the 3D mesh and point cloud of each individual part, as well as step-wise assembly instructions consisting of diagrams and text. The number of steps (i.e., the number of parts) for each assembly ranges from 2 to 20, with an average of 6.7 steps on average. Table 1 contrasts AssemblyBench with prior datasets proposed for assembly-related tasks. As is clear, AssemblyBench generalizes prior works while including ground-truth part assembly trajectories, exhibiting a variety of motion patterns. In Figure 2, we provide detailed statistics on the properties of our dataset. Out of all of the trajectories, 5.84% involve rotational movements such as sophisticated twists, 5.42% involve long translation movements that indicate insertions into a hole or slot, and there are about 58 parts that need long distance insertions combined with rotations for the assembly. We divide the set of 2789 assemblies into train/val/test splits in 80%-10%-10% allocation.

3.2. AssemblyBench Construction Pipeline

The following subsections present our automatic data generation pipeline of AssemblyBench, illustrated in Figure 3. We note that our pipeline is very general and could be used to generate assembly instruction manuals for a broad variety of objects, given only an object’s 3D CAD model. Such CAD models of assembled real-world objects are widely available (e.g., from machine designs).

Part Order and Motion Trajectories: There are two important sub-tasks in assembly: i) deciding the order of the parts to select for the assembly (so that it is physically re-

Table 1. Comparison of assembly/manipulation datasets. Legends:  diagrams,  text instructions,  videos,  3D objects.

Work	Domain	Input	Output	Size
IKEA-Manual [38]	Furniture	 	Assembly order, 6D part pose	102
IKEA-Manuals-at-Work [19]	Furniture	  	6D part pose, temporal alignments	36
ProMQA-Assembly [5]	Toys	  	Assembly order, General QA	78
LEGO-VLM [7]	Modular toy bricks	  	Object localization, state judgement	65
MEPNet [37]	Modular toy bricks	 	Assembly order, part pose	8000
ManualPA [43]	Furniture	 	Assembly order, 6D part pose	6871
CheckManual [20]	Appliances	  	Manipulation plan, action trajectory	369
Manual2Skill++ [34]	Furniture, toys, industrial	  	Assembly order, 6D pose trajectories	20+
AssemblyBench (Ours)	All the above except toy bricks	  	Assembly order, 6D pose trajectories	2789

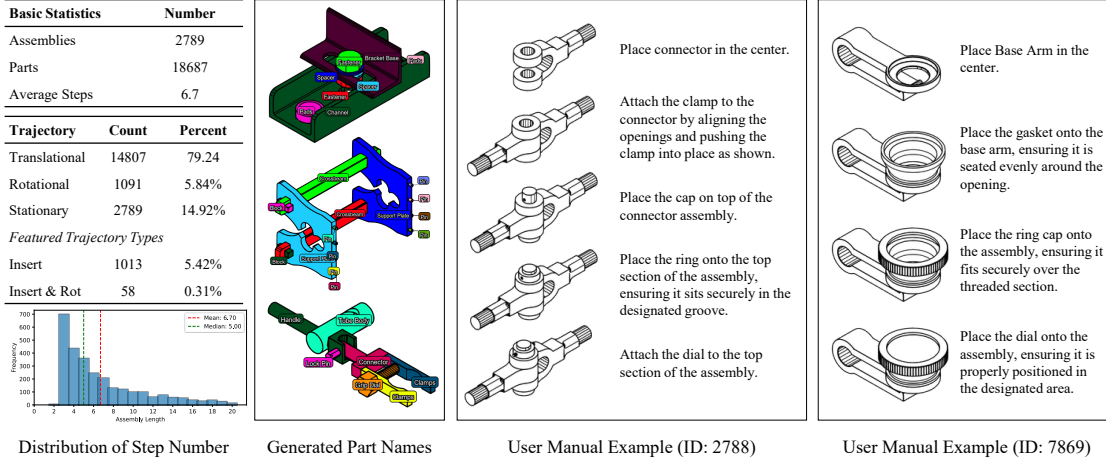


Figure 2. **Overview of AssemblyBench.** *Column 1:* Statistics of our AssemblyBench dataset and histogram of the number of parts per assembly in AssemblyBench. *Column 2:* Generated part names in AssemblyBench. The coloring and the labels are for visualization in this figure only—they are not included in the model inputs. *Columns 3–4:* Example generated instruction manuals for two different assemblies.

alizable) and ii) planning the motion of each part from its initial pose to the final assembled pose. The assembly-by-disassembly process [31, 32] tackles these sub-tasks via importing the 3D CAD models into a physics engine. Specifically, it uses a depth-first-search algorithm to attempt to disassemble the object one part at a time via applying forces along the part axes, until the part becomes disassociated from the other parts. This scheme discovers a disassembly sequence that includes both a disassembly order of parts and a 6-DoF pose trajectory for removing each part. Reversing both of these yields both the assembly part order and the assembly pose trajectories. To create AssemblyBench, we import each CAD object’s assembly steps and motion trajectories (discretized to T time steps) to Blender [6] and format them to produce assembly animations.

Diagram Generation: A key ingredient in the assembly process is the instruction manual, which any robotic platform intended to do assembly tasks must be equipped to follow for safety and physical feasibility. In AssemblyBench, for each assembly step and each camera view, we render the diagram using Blender as follows. First, we use a line-art style without coloring to mimic the visual style in real-world instruction manuals such as IKEA’s. Then, we use the

CAD part position at the final time step of the trajectory to represent the part’s final assembled state. We also render a segmentation map of this diagram for later text annotations.

We render the diagrams using a fixed set of isometric camera views. The diagrams from multiple camera views are used in two ways. First, they are selected in the later text annotation stage as the reference materials for a large vision-and-language model (VLM), as detailed later. Second, they are used to choose the camera view for the instruction manual, which uses a fixed camera view throughout all assembly steps (see Supplementary Material for details).

Instructional Text Generation:

To construct AssemblyBench, we generate text instructions to accompany each diagram, forming realistic step-by-step manuals. Our pipeline has two stages. First, we prompt a VLM (GPT-4.1) with diagrams of all individual parts to assign consistent names (e.g., “fastener”, “wire frame”), ensuring uniform terminology throughout the manual. Second, using these names, we prompt the VLM to produce textual instructions for each assembly step based on that step’s diagram.

Because our industrial assemblies contain complex shapes, frequent occlusions, and repeated part types (e.g.,

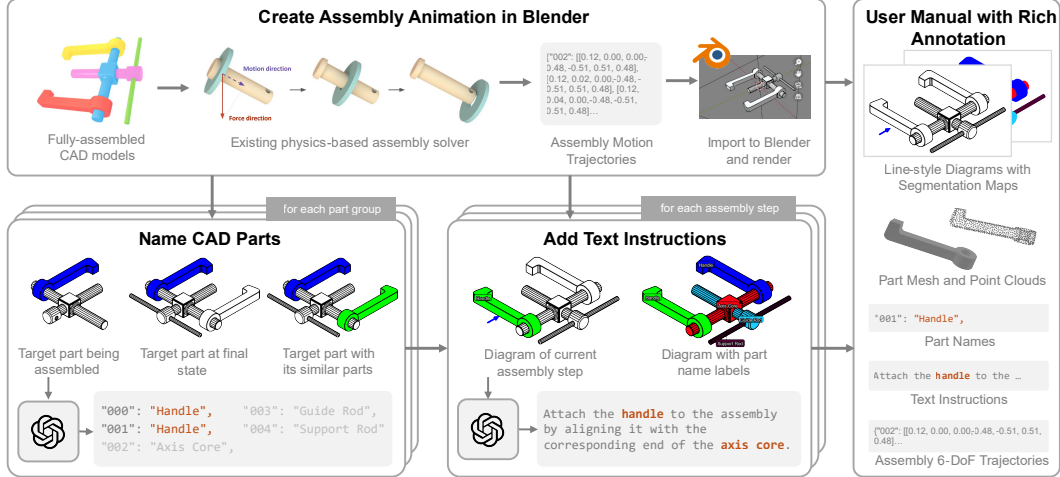


Figure 3. **Manual creation pipeline for AssemblyBench.** *Top left:* From a CAD model of an assembled object, we calculate the part assembly trajectories using a physical engine and import the animations to Blender. *Bottom left:* Blender renderings are fed to VLMs to create CAD part names and textual assembly instructions. *Right:* All annotations are used to generate a single step in the final manual.

multiple identical screws), obtaining consistent part names from a single camera view is challenging. Parts may be hidden in later steps (temporal occlusion), blocked from view (spatial occlusion), or duplicated. To address these issues, we apply several visual prompting techniques (described in the Supplementary Material). After generating consistent part names (see Fig. 2), we produce the step-by-step instructions by providing the VLM with: (i) the current step’s diagram with the target part highlighted, and (ii) the same diagram with all parts color-coded and labeled. This ensures consistent naming across all assembly steps (see “Add Text Instructions” in Fig. 3).

3.3. AssemblyDyno: Our Assembly Model

As shown in Figure 4, the input to AssemblyDyno is a step-by-step instruction manual and the corresponding set of separated 3D part point clouds. Our model starts with multimodal encoders, converting each instruction step and each part’s point cloud into feature embeddings of the same dimension D . After using an existing predictor to obtain part orders in the form of a permutation matrix, we apply a transformer decoder with positional encodings to predict the assembly trajectory for each step, as explained below.

Feature Extraction: We begin by applying off-the-shelf encoders to obtain semantic latent features from the inputs. For the 3D part point clouds $\{\mathcal{P}_i\}_{i=1}^N$, we use a lightweight PointNet variant [2], similar to that in [16, 43]. For the sequence of step diagrams $\{\mathcal{I}_j^{\text{img}}\}_{j=1}^N$, since the assembly instructions progress incrementally, we focus on the differences between successive steps. For any pair of consecutive diagrams $\mathcal{I}_j^{\text{img}}$ and $\mathcal{I}_{j+1}^{\text{img}}$, $j \in \{1, 2, \dots, N-1\}$, we form a difference image $|\mathcal{I}_j^{\text{img}} - \mathcal{I}_{j+1}^{\text{img}}|$ that highlights the newly added part relative to the partially assembled object. This difference image is then divided into K patches and

passed through a DINOv3 image encoder [28] to extract features. For the corresponding text content, we use the Qwen-3 embedding model [44] with frozen model weights. Each of the three modalities is projected to the same feature dimensionality D using linear layers, yielding part features $f^{\mathcal{P}} \in \mathbb{R}^{N \times D}$, image features $f^{\text{img}} \in \mathbb{R}^{N \times K \times D}$, and text features $f^{\text{txt}} \in \mathbb{R}^{N \times D}$. To fuse image and text features into one feature, we concatenate the two features by repeating text features along the patch dimension and apply linear projections, yielding instruction features $f^{\mathcal{I}} \in \mathbb{R}^{N \times K \times D}$.

Predicting Part Assembly Order: Applying a similar approach as Manual-PA [43], we calculate a similarity matrix between the embeddings of the parts and those of the instructions with a max-pooling operation on patch dimension K . Then, we use the Hungarian matching algorithm [13] to convert the similarity matrix into the predicted order $(\hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_N)$, which is projected into a permutation matrix $M \in \{0, 1\}^{N \times N}$.

Predicting Assembly Trajectories: To perform part-to-part interactions, we add the permutation matrix M with positional encoding [36] to the part features $f^{\mathcal{P}}$ and send the results to a self-attention transformer decoder. Then, we feed the outputs to a cross-attention module with temporal dimension, where position-encoded instruction features are added to produce the latent feature of assembly trajectories, shaped as $\mathbb{R}^{N \times T \times D}$. Finally, the latent feature is converted into a sequence of poses $\left\{ \left\{ (\hat{R}_i^k, \hat{t}_i^k) \right\}_{k=1}^T \right\}_{i=1}^N$ using a pose prediction head [43], where the rotations are represented using quaternions.

Training Losses: We train one model with the above architecture for part order prediction, and a second model with the same architecture for trajectory prediction. During trajectory prediction learning, we always feed the model using the ground-truth part order.

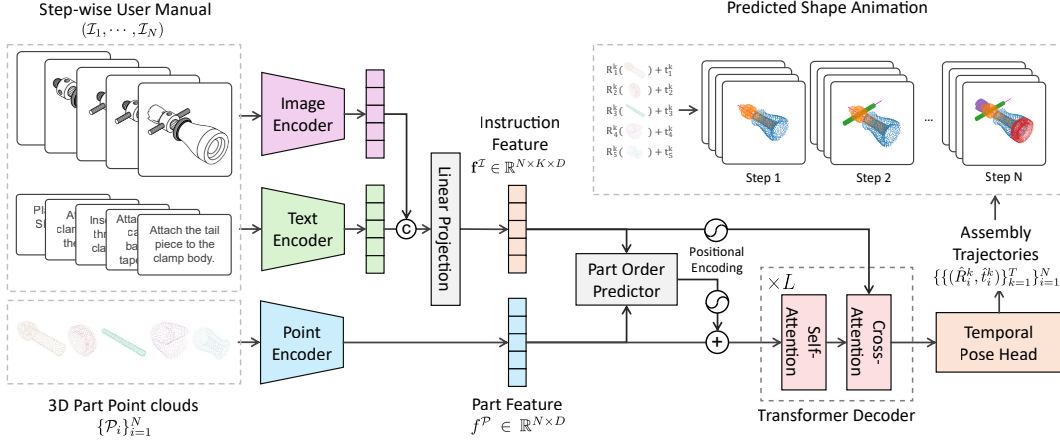


Figure 4. **Model Architecture of AssemblyDyNo.** (1) Feature Extraction: AssemblyDyNo starts with multi-modal encoders, converting user manual instructions and 3D part point clouds into embeddings of the same feature dimension D . (2) Predict Part Order: we use an existing predictor to get part order in the form of a permutation matrix. (3) Predict Assembly Trajectories: we use a transformer decoder with positional encodings to predict the assembly trajectory for each step.

Loss for Order Prediction: We optimize the similarity matrix between the instruction features f_j^I and part features $f_{\sigma(i)}^P$. In the similarity matrix, a correct match between f_j^I and $f_{\sigma(i)}^P$ yields a higher value. Based on this motivation, we adopt an InfoNCE loss [4, 23] design:

$$\mathcal{L}_{\text{order}} = -\frac{1}{B} \sum_{i=1}^B \log \frac{\exp(\text{sim}(\mathbf{f}_{\sigma(i)}^P, \mathbf{f}_i^I)/\tau)}{\sum_{j=1}^B \exp(\text{sim}(\mathbf{f}_{\sigma(i)}^P, \mathbf{f}_j^I)/\tau)},$$

where B is the batch size, $\sigma(i)$ denotes a permutation over part indices, $\text{sim}(\cdot)$ computes the similarity between two features, and τ is a temperature scaling factor.

Loss for Trajectory Prediction:

Inspired by [16, 43], the point-cloud loss \mathcal{L}_P measures the difference between the point clouds of the predicted final assembly at step T (all parts in their final poses) and the ground-truth assembly.

$$\mathcal{L}_P = \text{CD} \left(\bigcup_{i=1}^N (\hat{R}_i^{(T)} P_i + \hat{t}_i^{(T)}), \bigcup_{i=1}^N (R_i^{(T)} P_i + t_i^{(T)}) \right),$$

where $\bigcup_{i=1}^N$ represents the union of all N transformed parts to form the complete assembled shape, and $\text{CD}(\cdot)$ denotes the bidirectional chamfer distance, which measures the difference between two point clouds [3].

In addition to the overall point-cloud loss, we separately measure translation and rotation losses. For translation loss,

$$\mathcal{L}_T = \frac{1}{NT} \sum_{i=1}^N \sum_{k=1}^T \|\hat{t}_i^{(k)} - t_i^{(k)}\|_2,$$

where $\hat{t}_i^{(k)}$ is the predicted translation at time step k , $t_i^{(k)}$ is the ground-truth translation, and $\|\cdot\|_2$ denotes the ℓ_2 norm.

For rotational loss, as parts may have rotational symmetries, solely relying on ℓ_2 distance will miss some correct answers. So we use chamfer distance on the point clouds,

$$\mathcal{L}_R = \frac{1}{NT} \sum_{i=1}^N \sum_{k=1}^T \text{CD}(\hat{R}_i^{(k)} P_i, R_i^{(k)} P_i),$$

where $\hat{R}_i^{(k)}$ is the predicted rotation for part i at time step k , $R_i^{(k)}$ is the ground-truth rotation, and $\text{CD}(\cdot)$ denotes the bidirectional chamfer distance between the rotated point clouds.

To encourage temporally smooth motions, we additionally regularize the frame-to-frame “velocity” of both translations and rotations of the predicted trajectories. We penalize the finite difference between consecutive frames,

$$\mathcal{L}_{S_T} = \frac{1}{N(T-1)} \sum_{i=1}^N \sum_{k=1}^{T-1} \|\hat{t}_i^{(k+1)} - \hat{t}_i^{(k)}\|_2^2, \quad (1)$$

$$\mathcal{L}_{S_R} = \frac{1}{N(T-1)} \sum_{i=1}^N \sum_{k=1}^{T-1} \|\hat{q}_i^{(k+1)} - \hat{q}_i^{(k)}\|_2^2. \quad (2)$$

The final loss is a weighted sum of all the components:

$$\mathcal{L} = \lambda_P \mathcal{L}_P + \lambda_T \mathcal{L}_T + \lambda_R \mathcal{L}_R + \lambda_{S_T} \mathcal{L}_{S_T} + \lambda_{S_R} \mathcal{L}_{S_R}. \quad (3)$$

3.4. Physics-Aware Evaluation in AssemblyBench

Our evaluation captures three aspects of assembly: if a model: i) correctly grounds each diagram to its part, ii) can make correct prediction of the final 3D poses of the parts (*Static Pose Estimate*), and iii) can predict 3D assembly trajectories that are physically feasible (*Assembly in Simulator*). For (i), we use the standard Kendall’s Tau (KD) [10] metric to compute the correlation between the actual assembly order $(\pi_1, \pi_2, \dots, \pi_N)$ and the predicted one $(\hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_N)$. For (ii), similar to prior works,

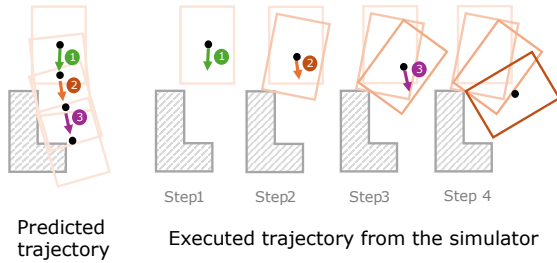


Figure 5. We use a physics simulator to execute the predicted assembly trajectory, evaluating whether it is physically feasible.

[16, 17, 43], we use the Shape Chamfer Distance (SCD), Part Assembly Correctness (PA), and Success Rate (SR). For (iii), we detail our new evaluation framework below.

As noted above, physical feasibility of predicted assembly trajectories is important for real-world adoption. In order to capture such feasibility, we propose to execute the predicted trajectories within a physics-based simulator to check their correctness. We use Newton Physics [22] as our simulator and evaluate the assembly process step-by-step: For each predicted step, we use the velocity sequence of the predicted trajectory as the control signal to roll out the simulation. Then we measure the difference between the simulation results and the ground truth.

Initially, we set up the parts that were assembled in previous steps of the instruction manual within the simulator, using their predicted final poses. Then we place the current part that is to be assembled, using the pose from the first time step of its predicted trajectory. We ignore gravity in this work for simplicity. Next, as shown in Figure 5, we let Δt represent the duration of each time step of the trajectory. Then we start the simulation by assigning v_1 as the initial velocity of the part and let the simulation run for Δt . The part might collide into other parts and change its velocity during that time step. After Δt , we iteratively assign v_2 and v_3 each for the same amount of time, and so on through to the final time step. The process is depicted in Figure 5.

We measure the prediction quality by comparing the executed pose trajectory from the simulator to the ground truth. Apart from using **PA** and **SR** in the simulation setting by following the same algorithms, we also present two new metrics to account for part symmetries. Specifically, due to the part symmetries and thus potential for non-unique solution poses, we do not compute the translation or rotational differences. Instead, we apply chamfer distance on two common trajectory evaluation metrics, i.e., ADE and FDE, resulting in the following variants:

- **Average Chamfer Distance (ACD)**: For each time step, we apply the corresponding executed pose and ground truth pose to the point clouds of their related parts and computes the chamfer distance between them. The chamfer distances from all time steps are averaged and the quartile values for all parts in all assemblies is reported.

- **Final Chamfer Distance (FCD)**: Similar to ACD, but instead of averaging over all time steps, we only take the chamfer distance of the final time step and report the median over all assemblies. We report the quartile values for all parts in all shapes.

4. Experiments

Table 2 provides a comprehensive evaluation of assembly trajectory prediction, final pose estimation, and full assembly simulation across multiple ablations. We compare AssemblyDyno to its ablations and to [43]. Since some baselines (e.g., [43]) do not predict trajectories, we generate them heuristically: from the predicted final pose, we translate the part outward from the object’s center of mass (without rotation) for a distance equal to half the diagonal of its predicted bounding-box. Reversing this path yields the assembly motion.

Part Order and Final Pose: We first analyze the *GT part-order* setting, which serves as a sanity check to isolate the model’s trajectory-prediction and pose-estimation capabilities from ordering errors. When provided with perfect part orders, AssemblyDyno consistently achieves the strongest performance across nearly all metrics. In the *Final Pose Estimate* block, AssemblyDyno attains the lowest SCD scores and the highest PA/SR values among the compared methods, highlighting its accurate trajectory reasoning and robust geometric understanding. In the *Assembly in Simulator* block, AssemblyDyno also yields superior PA and SR, even though the model is not trained with simulator feedback (see Figure 6). This demonstrates strong real-to-sim transfer of predicted motion trajectories. The ablation results further validate key model components: removing the text encoder (w/o text) or replacing the trajectory predictor with the naïve baseline (w/o trajectory) leads to consistent drops in PA, SR, ACD, and FCD, underscoring the importance of both textual grounding and the dedicated trajectory module.

We then analyze the *standard setting*, where the model must predict part orders before generating trajectories. Here, performance decreases across all methods, reflecting the substantial influence of ordering quality on downstream assembly outcomes. Despite this, AssemblyDyno still outperforms prior work, though the margin over its own ablations becomes smaller. This trend is especially visible in PA and SR, where the advantage of AssemblyDyno over w/o text narrows. We attribute this to two key factors: (1) assembly order prediction becomes the dominant bottleneck, and errors in ordering propagate into trajectory and pose predictions; and (2) textual information contributes limited additional signal for order prediction, since step-wise diagrams already strongly constrain the next operation, reducing the marginal benefit of language inputs at this stage. Overall, the table highlights three central insights: (i) our

Table 2. **Part assembly results on the test split of AssemblyBench.** We bold the best results and highlight the second best results in blue.

Model	Ordering	Final Pose Estimate				Assembly in Physics Simulator							
		KD \uparrow	SCD(10^{-3}) \downarrow	PA($\%$) \uparrow	SR($\%$) \uparrow	ACD(10^{-3}) \downarrow			FCD(10^{-3}) \downarrow			PA($\%$) \uparrow	SR($\%$) \uparrow
					25%	50%	75%	25%	50%	75%			
<i>Standard Setting</i>													
AssemblyDyno	0.819	3.91	71.21	34.64	6.95	34.77	156.74	3.31	15.97	110.30	42.76	13.57	
w/o text	0.805	3.83	70.28	35.00	6.99	33.68	148.86	3.03	17.25	99.77	41.77	15.00	
w/o trajectory	0.819	4.42	67.63	30.00	104.73	188.94	317.76	11.95	58.52	199.64	22.09	1.43	
ManualPA [43] (ICCV'25)	0.788	4.24	70.04	33.57	104.66	192.79	325.50	11.75	56.04	209.09	23.24	1.79	
<i>Use GT part orders</i>													
AssemblyDyno	–	3.87	79.69	44.29	3.23	9.97	29.74	1.50	4.43	12.41	70.15	33.57	
w/o text	–	3.78	78.19	42.86	3.45	10.77	35.28	1.61	4.85	14.32	67.96	31.79	
w/o trajectory	–	3.80	79.31	43.21	136.48	235.16	390.23	6.90	37.39	197.65	29.85	3.57	
ManualPA [43] (ICCV'25)	–	4.15	77.40	39.28	142.32	230.52	382.62	6.02	35.00	196.51	31.33	2.14	

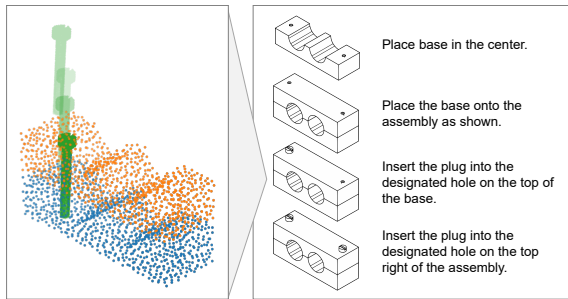


Figure 6. Given the instruction manual on the right, AssemblyDyno showcases the capability of predicting insertion assembly trajectory. Key frames of the trajectory are shown on the left.

model is intrinsically strong at pose and trajectory prediction, as shown by the GT-order results; (ii) the ablations validate the contributions of the text encoder and structured trajectory module; and (iii) in realistic deployment conditions, improving part-order prediction is crucial for unlocking further gains in downstream assembly performance.

Physics-based Evaluation:

We further show that our simulator-based evaluation provides a more stringent assessment of trajectory quality. Figure 7 compares the median translation errors of the *predicted* trajectories (orange) and the corresponding *simulated* trajectories (green). Although both decrease over time, the simulated error remains higher and the gap widens toward later frames, indicating that the simulator exposes compounding inaccuracies such as collisions or infeasible motions. Early in the trajectory, the simulated-error variance is larger because early-time-step predictions contains large variance that deviate substantially from ground truth. As the object approaches assembly, the physical constraints align more closely with the true configuration, reducing variance; however, these same constraints and obstacles still impede the predicted motion from reaching the final pose, resulting in a slightly higher but more stable residual error. Thus, the simulator-based metric offers a stricter and more realistic measure of trajectory feasibility.

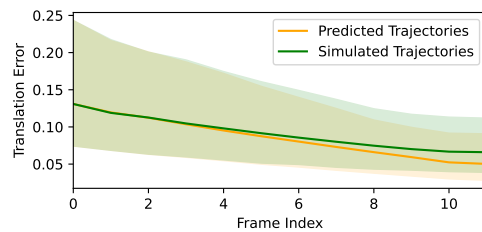


Figure 7. Median translation error with respect to ground truth as a function of trajectory frame. Shaded regions denote the 25th–75th percentiles computed across all object parts. The simulated trajectories exhibit consistently larger deviations due to collisions and other dynamic interactions, revealing hidden failure modes that are not captured when evaluating predictions alone.

5. Conclusion

We introduced AssemblyBench, a large-scale, multi-modal assembly dataset that extends beyond furniture to include complex industrial objects, complete with step-wise diagrams, textual descriptions, and ground-truth 6-DoF part trajectories. Building on this foundation, we presented AssemblyDyno, a unified transformer-based architecture that jointly predicts assembly order, final poses, and physically plausible motion trajectories. Our experiments demonstrate that existing state-of-the-art methods struggle on the richer and more challenging scenarios offered by AssemblyBench, while AssemblyDyno achieves substantially stronger performance in both final pose estimation and simulator-executed assembly. The analysis further highlights the benefits of integrating multi-modal manual information and structured trajectory prediction, as well as the importance of accurate order prediction for full assembly success. Overall, our work provides a comprehensive benchmark and modeling framework that brings instruction-guided assembly significantly closer to real-world applicability, with opportunities for future advances in order prediction, physical reasoning, and robotic execution.

Please see the supplementary material for more detailed results.

References

- [1] Yizhak Ben-Shabat, Xin Yu, Fatemeh Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 847–859, 2021. 2
- [2] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 5
- [3] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 6
- [4] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006. 6
- [5] Kimihiro Hasegawa, Wiradee Imrattanastrai, Masaki Asada, Susan Holm, Yuran Wang, Vincent Zhou, Ken Fukuda, and Teruko Mitamura. Promqa-assembly: Multimodal procedural qa dataset on assembly. *arXiv preprint arXiv:2509.02949*, 2025. 3, 4
- [6] Roland Hess. *The essential Blender: guide to 3D creation with the open source suite Blender*. No Starch Press, 2007. 4
- [7] Haochen Huang, Jiahuan Pei, Mohammad Aliannejadi, Xin Sun, Moonisa Ahsan, Chuang Yu, Zhaochun Ren, Pablo Cesar, and Junxiao Wang. Lego co-builder: exploring fine-grained vision-language modeling for multimodal lego assembly assistants. *arXiv preprint arXiv:2507.05515*, 2025. 4
- [8] Youngkyoon Jang, Brian Sullivan, Casimir Ludwig, Iain D. Gilchrist, Dima Damen, and Walterio Mayol-Cuevas. Epicent: An egocentric video dataset for camping tent assembly. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4461–4469, 2019. 2
- [9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. 3
- [10] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938. 6
- [11] Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. In *Advances in Neural Information Processing Systems*, pages 7552–7579. Curran Associates, Inc., 2024. 2
- [12] Zachary Kingston, Mark Moll, and Lydia E. Kavraki. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(Volume 1, 2018):159–185, 2018. 3
- [13] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 5
- [14] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. 3
- [15] Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. CAD-Llama: Leveraging large language models for computer-aided design parametric 3d model generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18563–18573, 2025. 2
- [16] Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas Guibas. Learning 3d part assembly from a single image. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI*, page 664–682, Berlin, Heidelberg, 2020. Springer-Verlag. 1, 3, 5, 6, 7
- [17] Yulong Li, Andy Zeng, and Shuran Song. Rearrangement planning for general part assembly. In *Proceedings of The 7th Conference on Robot Learning*, pages 127–143. PMLR, 2023. 3, 7
- [18] Yichen Li, Kaichun Mo, Yueqi Duan, He Wang, Jiequan Zhang, Lin Shao, Wojciech Matusik, and Leonidas Guibas. Category-level multi-part multi-joint 3D shape assembly. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3281–3291, 2024. 3
- [19] 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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. 2, 3, 4
- [20] Yuxing Long, Jiyao Zhang, Mingjie Pan, Tianshu Wu, Tae-whan Kim, and Hao Dong. CheckManual: A new challenge and benchmark for manual-based appliance manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 4
- [21] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [22] Newton Contributors. Newton: GPU-accelerated physics simulation for robotics, and simulation research., 2025. 2, 7
- [23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 6
- [24] Mayank Patel, Rahul Jain, Asim Unmesh, and Karthik Ramani. Dynamo: Dependency-aware deep learning framework for articulated assembly motion prediction. *2509.12430*, 2025. 2
- [25] Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14798–14809, 2025. 2

- [26] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis and composition. In *IEEE Proceedings of the International Conference on Computer Vision, (ICCV)*, pages 8758–8767. IEEE, 2019. Publisher Copyright: © 2019 IEEE.; 17th IEEE/CVF International Conference on Computer Vision, ICCV 2019 ; Conference date: 27-10-2019 Through 02-11-2019. 3
- [27] Fadime Sener, Dibyadip Chatterjee, Daniel Sheleпов, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21096–21106, 2022. 2
- [28] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 5
- [29] Daniel Sliwowski, Shail Jadav, Sergej Stanovcic, Jędrzej Orbik, Johannes Heidersberger, and Dongheui Lee. Reassemble: A multimodal dataset for contact-rich robotic assembly and disassembly. *arXiv preprint arXiv:2502.05086*, 2025. 2
- [30] Bingjie Tang, Iretoiyo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S. Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024. 1
- [31] Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D.D. Willis, and Wojciech Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Trans. Graph.*, 41(6), 2022. 2, 3, 4
- [32] 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)*, pages 4380–4386. IEEE, 2024. 2, 3, 4
- [33] Yunsheng Tian, Joshua Jacob, Yijiang Huang, Jialiang Zhao, Edward Gu, Pingchuan Ma, Annan Zhang, Farhad Javid, Branden Romero, Sachin Chitta, et al. Fabrica: dual-arm assembly of general multi-part objects via integrated planning and learning. *arXiv preprint arXiv:2506.05168*, 2025. 1
- [34] Chenrui Tie, Shengxiang Sun, Yudi Lin, Yanbo Wang, Zhongrui Li, Zhouhan Zhong, Jinxuan Zhu, Yiman Pang, Haonan Chen, Junting Chen, et al. Manual2Skill++: Connector-aware general robotic assembly from instruction manuals via vision-language models. *arXiv preprint arXiv:2510.16344*, 2025. 1, 2, 3, 4
- [35] 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. 1, 2, 3
- [36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5
- [37] 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 (ECCV)*, pages 677–694. Springer Nature Switzerland, 2022. 2, 4
- [38] Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Ran Zhang, Chin-Yi Cheng, and Jiajun Wu. IKEA-Manual: Seeing shape assembly step by step. In *NeurIPS 2022 Datasets and Benchmarks Track*, 2022. 1, 2, 3, 4
- [39] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. PQ-NET: A generative part seq2seq network for 3d shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3
- [40] Boshen Xu, Sipeng Zheng, and Qin Jin. SPAFormer: Sequential 3D part assembly with transformers. In *International Conference on 3D Vision (3DV)*, pages 1317–1327, 2025. 3
- [41] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. CAD-MLLM: Unifying multimodality-conditioned CAD generation with MLLM. *arXiv preprint arXiv:2411.04954*, 2024. 2
- [42] Jiahao Zhang, Anoop Cherian, Yanbin Liu, Yizhak Ben-Shabat, Cristian Rodriguez, and Stephen Gould. Aligning step-by-step instructional diagrams to video demonstrations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3
- [43] Jiahao Zhang, Anoop Cherian, Cristian Rodriguez, Weijian Deng, and Stephen Gould. Manual-PA: Learning 3d part assembly from instruction diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6304–6314, 2025. 1, 2, 3, 4, 5, 6, 7, 8
- [44] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. 5
- [45] Hao Zheng, Regina Lee, and Yuqian Lu. Ha-vid: a human assembly video dataset for comprehensive assembly knowledge understanding. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2023. Curran Associates Inc. 2
- [46] Xinghao Zhu, Devesh K Jha, Diego Romeres, Lingfeng Sun, Masayoshi Tomizuka, and Anoop Cherian. Multi-level reasoning for robotic assembly: From sequence inference to contact selection. In *2024 IEEE international conference on robotics and automation (ICRA)*, pages 816–823. IEEE, 2024. 2, 3