2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multi-level Reasoning for Robotic Assembly: From Sequence Inference to Contact Selection

Xinghao Zhu[1,2], Devesh K. Jha[2], Diego Romeres[2], Lingfeng Sun[1], Masayoshi Tomizuka[1], Anoop Cherian[2]

***Abstract*— Automating the assembly of objects from their parts is a complex problem with innumerable applications in manufacturing, maintenance, and recycling. Unlike existing research, which is limited to target segmentation, pose regression, or using fixed target blueprints, our work presents a holistic multi-level framework for part assembly planning consisting of part assembly sequence inference, part motion planning, and robot contact optimization. We present the Part Assembly Sequence Transformer (PAST) – a sequence-to-sequence neural network – to infer assembly sequences recursively from a target blueprint. We then use a motion planner and optimization to generate part movements and contacts. To train PAST, we introduce D4PAS: a large-scale *Dataset for Part Assembly Sequences* consisting of physically valid sequences for industrial objects. Experimental results show that our approach generalizes better than prior methods while needing significantly less computational time for inference. Further details on our experiments and results are available in the video.**

I. Introduction

The assembly of parts in accordance with a target blueprint presents a compelling research frontier in the domains of robotics and machine learning. This task not only represents a highly valuable functionality that autonomous robots can perform, but it also embodies a complex problem space characterized by indeterminate intricacies. Achieving successful assembly requires robots to master several complex skills: understanding part geometries, reasoning about physical interactions and collisions, and executing assembly plans with robust sensing capabilities. To navigate through these complexities, robots need to cultivate an array of diverse competencies conducive to a successful assembly process. These include deciphering the assembly sequence, coordinating part trajectories, and identifying points of contact and their physical execution. Equally important is the robot's ability to generalize these competencies across various assemblies.

Previous investigations in robotics and computer vision have tackled this multifaceted challenge of part assembly through diverse methodological lenses. For example, one line of research attempts to sidestep the complexities of physics [1], electing to focus on more specialized tasks, such as the segmentation of target blueprints [2], [3] or the estimation of part poses [4], [5]. A second vein of research emphasizes the importance of physical interactions, specializing in the assembly of predetermined targets [6]–[9]. A third category adopts different target blueprints but imposes simplifying assumptions on part geometries (i.e., blocks) [10] or restricts its scope to a seen set of blueprint categories (e.g., chairs) [11]. The use of reinforcement learning (RL) has seen success in this space [10], [11]. However, RL-based solutions face challenges in terms of computational resources and efficiency. For instance, Ghasemipour et al. [10] requires an elaborate computational infrastructure involving thousands of CPUs and billions of steps for training, raising concerns about the practicality of the system. Our work aims to advance the field of robotic assembly by holistically considering intricate physical interactions between parts and designing a supervised training paradigm, while our approach is applicable to a broad spectrum of practical target blueprints.

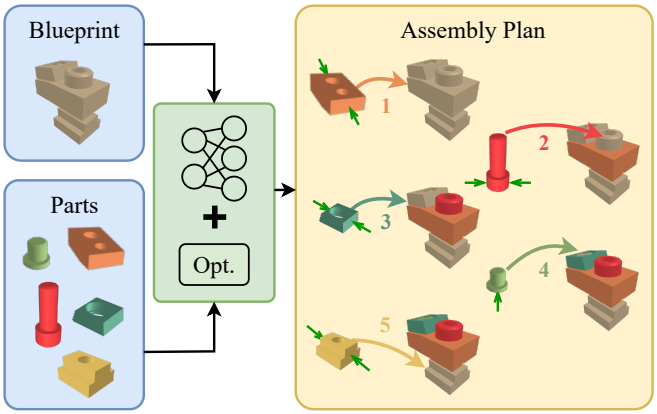

Fig. 1: Our goal is to facilitate robotic assembly across different target blueprints. Utilizing point clouds from target blueprints and assembly parts, our method identifies feasible assembly sequences (indicated by colored numbers), orchestrates part motions (represented by colored long arrows), and pinpoints contact points (denoted by short green arrows).

To achieve successful robotic assembly, this study breaks down the task into three distinct and key sub-tasks: 1) inferring the sequence in which the parts should be assembled, guided by the target blueprint, part shapes, and assembled poses; 2) coordinating the movements of the individual parts; and 3) identifying viable contact points for robotic manipulation. An illustration of these steps is provided in Fig. 1. Addressing the first challenge involves contending the physical interactions between parts and the inherent ambiguities. The former is due to the collision between parts that prevent arbitrary assembly order, and the latter arises due to multiple viable assembly sequences. Can we learn the order of assembling the parts statistically from their geometry and their locations in the target assembly? For example, it is

[1]Mechanical Systems Control Lab, UC Berkeley, Berkeley, CA, USA {zhuxh, lingfengsun, tomizuka}@berkeley.edu
[2]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA {jha, romeres, cherian}@merl.com

clear in Fig. 1 that the red screw can only be inserted if the orange piece is in place – the target blueprint and individual pieces collectively establish a specific order for assembly. We use this insight towards designing an implicit neural planning network using Transformers [12], dubbed *Part Assembly Sequence Transformer* (PAST) that takes as input point clouds of the target blueprint and the assembled parts and identifies the next parts to be assembled. Then, it is applied to generate the full sequence in an autoregressive fashion. For training our PAST model, we construct a benchmark dataset for part assembly sequences, dubbed D4PAS, by enumerating feasible assembly sequences [13].

To solve the problem of part motion planning for assembly, we leverage the RRT-connect [14] to generate trajectories from each part's resting pose to its assembled pose. Concurrently, we conduct an efficient physics-inspired multi-scale optimization of potential contact points on the part's surface to identify those that are most effective in achieving the desired part movement. Upon generating the assembly plan through the aforementioned steps, prior research has explored the use of reinforcement learning [15]–[17], model-predictive control [18], and diffusion policies [19] for its execution. However, the focus of this work is not on physical execution, which is earmarked for future investigation.

In summary, our primary contributions are as follows:

- We present an assembly planning algorithm to generate feasible part assemblies based on target blueprints, including inference of assembly sequences, planning of part movements, and optimization of contact points.
- We introduce the Part Assembly Sequence Transformer (PAST) to infer assembly sequence in an autoregressive fashion. PAST is designed to generalize to novel, diverse, and practical blueprints and part geometries.
- We provide a dataset for part assembly sequences (D4PAS), replete with assembly trajectories, enumerated assembly sequences, and viable contact points, thereby providing a foundation for future studies in robotic assembly.

II. Related Works

Previous research on **part assembly** varies in focus. On the one hand, approaches like [2], [3] tackle the problem via blueprint segmentation and part pose regression, employing point classification and optimization. Huang et al [4] use dynamic graph networks for geometric learning, while [5] utilize a shape-mating discriminator for shape-mating tasks. On the other hand, reinforcement learning (RL) has shown promise in physical assembly executions [6]. However, these RL methods often focus on fixed or single-category targets [7], [8], [10], [11] and entail costly training [10], [20].

To simplify part assembly, some works that streamline **assembly sequence inference** focus on optimizing the assembly order of individual parts [21]. Techniques leveraging precedence relationships in CADs have been used [22]. The assembly-by-disassembly strategy has been highlighted for its efficiency [13], [23]. In cases where parts are rigid, this strategy simplifies planning by reversing disassembly sequences [24], leveraging the bijection between assemblies and disassemblies. However, these methods often rely on time-consuming physical simulations. To address this, our work introduces the Part Assembly Sequence Transformer (PAST) for efficient sequence reasoning and geometrical understanding of parts and blueprints. We train the network using a dataset, D4PAS, generated from GPU-based simulations [25], employing assembly-by-disassembly techniques.

After sequence determination, robots require further guidance on **part motions and contact points** for efficient and robust execution. Works by [17], [26] demonstrate the importance of object movements and contact locations for robotic manipulations. For part motion planning, sampling-based methods like Rapidly-exploring Random Trees (RRT [14]) have proven effective in robotic motion planning. Zhu et al. [27] shows that contact optimization reliably identifies contact points for dexterous manipulation tasks. Building on these insights, our work fulfills part assembly with part motion generation and contact points identification, leaving the physical execution for future research.

III. Problem Overview

This work employs a part assembly formulation consistent with [11], [13], as shown in Fig. 1. Given M part meshes $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^{M}$ and their respective 6D assembled poses in the target blueprints $p^{tgt} = \{p_i^{tgt}\}_{i=1}^{M}$, the algorithm plans the assembly trajectories $(p^0, p^1, ...)$ for each part from their resting poses $p^0 = \{p_i^0\}_{i=1}^{M}$. We use p_i^t to represent the pose of part $\mathcal{M}_i$ at time t and use p^t to represent poses of all parts at time t. The algorithm assumes, at each time step, that only one part is in motion while the others remain stationary [13]. The work breaks down the complex task of assembly planning into three sub-tasks: assembly sequence inference, part motion planning, and contact point selection.

For one possible assembly, let the assembly sequence be denoted by $m = (m_k)_{k=1}^{M}$, specifying the order of part assembly. Each m_k belongs to the set $\mathcal{M}$ and identifies the kth part to be assembled. Part movements are represented by $\mathcal{T} = (p_{m_k})_{k=1}^{M}$, where p_{m_k} details the trajectory of parts when part m_k is moving throughout its moving horizon. Contact points facilitating these movements p_{m_k} are indicated by $\mathcal{C} = (c_{m_k})_{k=1}^{M}$. The assembly planning problem is formulated as:

$$\begin{aligned}\mathbb{P}(p^0, p^1, ..., p^{tgt}) =& \mathbb{P}(m, \mathcal{T}, \mathcal{C}) \\ =& \mathbb{P}(m) \cdot \mathbb{P}(\mathcal{T}|m) \cdot \mathbb{P}(\mathcal{C}|m, \mathcal{T}),\end{aligned}$$

where in this work, we assume a multi-level solution approach by sequentially solving for the sub-problems of (i) assembly inference, (ii) motion planning, and (iii) contact selection, in that order.

It is crucial to recognize that feasible assembly sequences are a subset of all possible part permutations. This constraint arises from the potential for part collisions, which precludes arbitrary assembly sequences. For instance, a washer must be in place before tightening a screw. To address this combinatorial inference problem approximately, we introduce the Part Assembly Sequence Transformer (PAST) to learn statistical

correlations between the parts and the target blueprint to produce physically viable assembly sequences $\mathbb{P}(m)$. The network ingests both target blueprints and unassembled parts, outputting the next feasible parts for assembly. This iterative process determines the full assembly sequence, as in Fig. 2.

Upon establishing the assembly sequence m, part movement can be planned using established motion planning algorithms $\mathbb{P}(\mathcal{T}|m)$. Following that, the algorithm optimizes the contact points that the robot can utilize to execute these movements, $\mathbb{P}(\mathcal{C}|m, \mathcal{T})$, thus culminating in a coherent assembly process.

IV. Assembly Planning

The preceding section outlines our multi-level approach to assembly planning. This section delves into the details of each of the three planning levels.

A. Part Assembly Sequence Transformer (PAST)

The transformer ingests the target assembly blueprint and the remaining unassembled parts, outputting a probability for each part's suitability for assembly at the current step. The part with the highest probability is chosen for assembly and removed from the list of remaining parts. This iterative process continues until all parts are assembled, yielding one assembly sequence $m = (m_k)_{k=1}^{M}$. Fig. 2 illustrates this recursive planning approach.

PAST takes two branches of inputs: a target assembly blueprint and unassembled remaining parts, both represented using point clouds. For an assembly comprising M parts, the target blueprint is rendered with the 6D assembled poses p^{tgt} of all parts, resulting in a point cloud $PC_{tgt} \in \mathbb{R}^{N_t \times 6}$. This point cloud consists of N_t sampled points, each with positional and normal features. During the kth step of assembly, PAST selects the next part to assemble from the $M-k$ remaining parts. These remaining parts are input as $M-k$ individual point clouds, denoted as $\{PC_{r,i}\}_{i=1}^{M-k}$, where $PC_{r,i} \in \mathbb{R}^{N_r \times 6}$ and contains N_r sampled points.

A key design question for PAST is which neural model to use for representing the input point clouds. Among point cloud encoders, such as PointNet and its variants [28], [29], it was shown in [5] that dynamic graph CNN (DGCNN) [30] offers superior efficiency and representational capabilities in assembly segmentation. To this end, we employ DGCNN to derive target features $v \in \mathbb{R}^{N_t \times h}$ from the target blueprint (one feature for every sample point) and part features $u_i \in \mathbb{R}^h$ from each remaining part (i.e., one feature for every part after max-pooling the features from all samples belonging to that part). Here, h is the hidden feature dimension and $i \in \{1, ..., M-k\}$.

Once the features are extracted, our PAST model then jointly refines these features through L transformer blocks, as illustrated in Fig. 2. As is well-known, transformers use self- and cross-attention to learn correlations between their inputs and have demonstrated state-of-the-art performances in generating sequential outputs (e.g., language). Our key insight is to use such attention to learn physically plausible assembly sequences in a supervised setting. Mathematically, suppose for a *query* set $\mathbf{q}$ and a *key* set $\mathbf{k}$, let

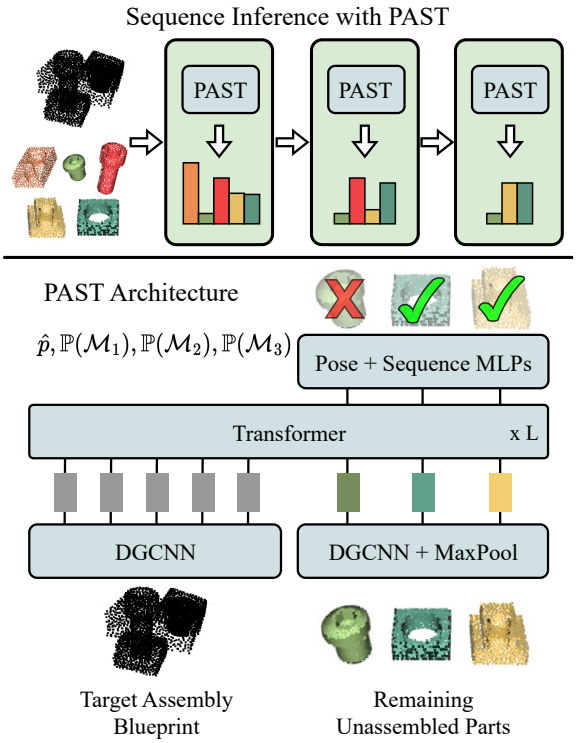

Fig. 2: Sequence inference pipeline and PAST architecture. (Top) PAST operates sequentially to estimate the assembly probability $\mathbb{P}(\mathcal{M}_i)$ for each remaining part. The part with the highest probability is chosen for assembly. (Bottom) Using the third block as an example, PAST selects one part for assembly from the three remaining options. PAST also performs pose regression for each part $\hat{p}$ as an auxiliary task.

the transformer dot-product attention[1] operator be defined as $\text{Attention}(\mathbf{q}, \mathbf{k}) = W_v(\mathbf{k})^T \text{softmax}\left(\frac{W_k(\mathbf{k})W_q(\mathbf{q})}{\sqrt{h}}\right)$, where W_q, W_k, W_v are matrices embedding the query and the key sets in a common latent space. Our PAST transformer blocks utilize a two-stage approach for feature refinement between the target and parts. The first stage independently processes and updates the features with self-attention; that is,

$$v = \text{Attention}(v, v) \text{ and } u_i = \text{Attention}(U, u_i),$$

where $U = \{u_i\}_{i=1}^{M-k}$ denotes all part point cloud features. The second stage applies cross-attention between the target features and part features [31], updating them to:

$$\hat{v} = \text{Attention}(U, v) \text{ and } \hat{u}_i = \text{Attention}(v, u_i).$$

where $\hat{v}$ and $\hat{u}_i$ are fed into the next block.[2] In the final step, PAST calculates the assembly probability for each part using the formula $\mathbb{P}(\mathcal{M}_i) = \text{MLP}(u_i)$. The part to be assembled next is then selected based on the maximum probability, denoted as $m_k = \text{argmax}_i \, \mathbb{P}(\mathcal{M}_i)$. In addition to predicting the assembly sequence, the transformer also estimates the 6D assembled pose p_i^{tgt} for each part as an

[1]For simplicity of notation, we have avoided details on multi-head attention and other processing modules in the transformer.

[2]We use the same attention expression for all blocks except for replacing u and v updated to $\hat{u}$ and $\hat{v}$ from the previous block.

auxiliary task, represented as $\hat{p}_i = \text{MLP}_p(u_i)$. The inclusion of this auxiliary task enhances the network's capability to comprehend the geometric interrelations between parts, a technique that has proven effective in [2], [5].

We use supervised learning for training PAST via estimating the predicted assembly probability $\mathbb{P}(\mathcal{M}_i)$ with mean squared error loss $\text{MSE}(y_i, \mathbb{P}(\mathcal{M}_i))$ with the feasibility of assembling a part $\mathcal{M}_i$ at a given step is denoted y_i. Additionally, the pose regression task aims to minimize the difference between the actual 6D assembled pose p_i^{tgt} and the predicted pose $\hat{p}_i$. The difference is calculated with $\sum_i \left\|p_{i,tra}^{tgt} - \hat{p}_{i,tra}\right\| + \left\|p_{i,rot}^{tgt} - \hat{p}_{i,rot}\right\|$, where $p_{i,tra}^{tgt}, \hat{p}_{i,tra}$ indicate target and predicted translation for each part and $p_{i,rot}^{tgt}, \hat{p}_{i,rot}$ represent the axis-angle.

B. Dataset for Part Assembly Sequences (D4PAS)

To train PAST, we introduce a new *dataset for part assembly sequences* or D4PAS. Each sample in the assembly sequence dataset comprises multiple components, namely: (i) the target blueprint point cloud PC_{tgt}, (ii) point clouds for $M-k$ remaining parts $\{PC_{r,i}\}_{i=1}^{M-k}$, and (iii) the feasibility of assembly for each part $\{y_i\}_{i=1}^{M-k}$. The feasibility y_i specifies whether a part $\mathcal{M}_i$ can be assembled at the current step and can subsequently lead to a successful final assembly. Note that there can be many viable part candidates at every step, derived from all possible sequence enumerations using the scheme in assembly-by-disassembly [13], as described in Algorithm 1 and illustrated in Fig. 3.

Algorithm 1 Disassembly Planning

1: **Input:** part meshes $\{\mathcal{M}_i\}_{i=1}^{M}$ and inertias $\{I_i\}_{i=1}^{M}$
2: **Input:** target part pose p^{tgt}, empty queue $\mathcal{J}$
3: **Output:** sequence of disassembly
4: $\mathcal{J}$.enq(p^{tgt}, f_i^j) for $f_i^j \propto I_i$ and $i \in \{1, \cdots, M\}$
5: **while** not finish **do** ▷ In parallel
6: $(p^t, f_i^j) = \mathcal{J}$.deq ▷ BFS
7: **if** success(p^t) **then**
8: return GetSequence(p^{tgt}, p^t)
9: **end if**
10: $p^{t-1} =$ simulate(p^t, f_i^j) ▷ Disassembly attempt
11: **if** isNovel(p^{t-1}) and isExec(p^t, p^{t-1}) **then**
12: $\mathcal{J}$.enq(p^{t-1}, f_i^j) for all unassembled part i
13: **end if**
14: **end while**

The disassembly planning algorithm operates in a search-based manner, where the set of parts poses' at time t, $p^t = \{p_i^t\}_{i=1}^{M}$, serves as the search state. At each step, the algorithm selects an unassembled part i (line 6) and attempts to remove it from the blueprint with a physical simulation (line 10). The selection of the moving part follows a breadth-first search (BFS) scheme, ensuring a comprehensive enumeration of all possible disassembly sequences. The chosen part i is moved in the direction of its moment of inertia I_i with force f_i^j, calculated as $f_i^j = \Lambda_i e_j I_i$, where Λ_i is the mass of the part and e_j is a one-hot vector with nonzero element at $j \in \{1, 2, 3\}$. Torque is also applied to enable rotational movement and is computed similarly to f_i^j. The search queue is expanded only if the disassembly attempt results in novel part poses and is executable by a robot (line 11). Part poses are regarded as a novel (isNovel) if they lead to a new location for one of the parts. The feasibility of execution (isExec) is confirmed by determining whether there exists collision-free grasp or push points on the part's surface. After each disassembly attempt, the remaining unassembled parts are added back to the queue, considering all possible dragging forces (line 12).

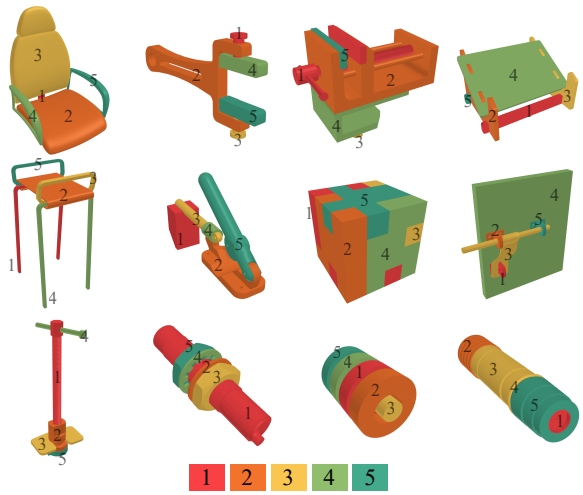

Fig. 3: Example assembly sequences in our dataset. For each target blueprint, we enumerate all feasible assembly sequences and present one representative sequence here. The color coding and the numbers beside each part signify the assembly sequence, as indicated by the color bar.

Compared to [13], the dataset generation algorithm in this work incorporates several advancements. First, we employ a parallel simulation (Isaac Gym [25]) to expedite the dataset generation and allow future studies on assembly planning with RL [10]. Second, unlike [13] that use arbitrary force directions, our algorithm applies disassembly forces along the part's moment of inertia, aligning with the physical properties of the parts. Third, we incorporate additional constraints to ensure that the generated plans are executable by robots (isExec), making our dataset more practical for further robotic applications.

C. Part Motion and Contact Planning

Once the sequence is determined by the PAST transformer, the next steps involve planning the movements of the parts and identifying suitable contact points for robotic manipulation. For part movements, we use the RRT-connect [14] to search for a collision-free path for each part, in line with the inferred assembly sequence.

For contact points planning, this work first plans robust grasps for each part with two contact points and filters out those in collision with other parts [32], [33]. To do this, we enumerate all grasp pairs from the part point cloud $PC_{r,i}$ and use the Ferrari Canny metrics to determine their robustness [34]. We then identify the feasibility of execution

using FCL [35] to check the collision between grasp points and the assembled parts along the assembly trajectory. If no feasible grasps are found, we resort to optimizing for a single pushing point c by solving the optimization [27]:

$$\begin{aligned} \min_{c,F_c} \quad & F_c \\ s.t. \quad & F_c \in FC(c) \\ & G(c)F_c = W \end{aligned} \tag{1}$$

where F_c is the pushing force at c. $FC(c)$ is the friction cone at point c and is expressed by $F_{c,1}^2 + F_{c,2}^2 \leq \mu F_{c,3}^2$ with μ being the friction coefficient. $G(c) \in \mathbb{R}^{6\times 3}$ is the grasp map which maps the contact force to the part movement force [36]. W is the part movement force derived from the part movement [27]. We observe that the optimization problem (1) can be cast as a semi-definite program (SDP) once the pushing point c is specified. To solve this problem, we employ a hybrid approach that combines sampling with an SDP solver [27]: the pushing point c is sampled from the part's point cloud and held constant during the optimization of F_c as SDP. The outcomes are iteratively updated across all sampled pushing points to identify the optimal solution.

V. Experiments

This section offers details of our model, dataset, and empirically validates our approach against the related methods.

A. Dataset and Network Details

Both disassembly planning and PAST training were performed using a single RTX3090 GPU. During the disassembly planning, properties of parts Λ_i, I_i were extracted from the simulation. The friction $\mu = 0.2$. Our dataset comprises 8,670 target blueprints and a total of 84,326 assembly sequences, broken down as follows: 7,278 are 3-step sequences, 54,612 range from 4 to 7 steps, and 22,436 have more than seven steps. These sequences can be further augmented with varying choices of the remaining (unassembled) parts. Specifically, for an assembly sequence with M parts, we randomly split the sequence into two segments with size k and $M-k$, respectively. Parts in the second segment are regarded as unassembled, and the first part in the second segment is the part that can be assembled with $y_{M-k} = 1$. Additionally, we gather segments from all viable sequences to identify all potential parts that can be assembled for a specific unassembled segment.

During the training of PAST, two-part assemblies were only used to train the auxiliary pose regression. The target blueprint was sampled at $N_t = 1024$ points, and each part mesh was sampled at $N_r = 512$ points. The DGCNN encodes point clouds with dimension $h = 256$. PAST uses $L = 8$ transformer blocks and was implemented in PyTorch and was trained with the AdamW optimizer using the One-Cycle learning rate scheduler. The target blueprints were re-centered and normalized to a unit ball and randomly rotated and jittered as augmentation. We allocate 500 multi-part assemblies for the test set, reserving others for training.

TABLE I: Quantitative results on assembly sequence inference. We report three metrics: one-step prediction accuracy (`1-Acc`), full sequence prediction accuracy (`Seq-Acc`), and the computation time (CT).

	`1-Acc` (%)	`Seq-Acc` (%)	CT (ms)
NSM [5]	75.0	57.8	58.5
DGL [37]	77.4	54.1	104.4
ATA [13]	NA	NA	24312.6
Seg-PAST	90.3	80.4	**52.2**
NoAux-PAST	79.1	58.4	**52.2**
PAST	**91.7**	**82.9**	**52.2**

B. Baselines and Ablations

The baseline and ablation studies aim to evaluate the efficacy of PAST and the overall algorithm.

Metrics. We employ two key metrics to assess the performance of assembly sequence inference: one-step prediction accuracy (`1-Acc`) and sequence prediction accuracy (`Seq-Acc`). A one-step prediction is deemed correct if the selected part, sampled based on predicted assembly probabilities, belongs to the set of possible parts for assembly. In addition, we apply PAST in an autoregressive fashion to generate a full assembly sequence, which is considered correct if it aligns with any of the possible assembly sequences for that object in our dataset. In addition, we also report the computational time (CT) taken for full sequence inference.

Compared algorithms. Given that no existing solutions are tailored specifically for part assembly sequence inference, we adapt methods from part segmentation and pose regression as baselines, and compare against various ablations.

- NSM (Neural Shape Mating [5]) uses a transformer to address two-part shape mating. We adapt this network to accommodate multiple part inputs.
- DGL (Dynamic Graph Learning [37]) employs a graph neural network to perform assembly pose regression [38]. We use a global node to represent target assembly [39] for assembly sequence inference.
- ATA (Assemble-Them-All [13]) solves assembly through runtime physical simulation. The assembly process aligns with Algorithm 1.
- Seg-PAST: We substitute the final pose regression layer in PAST to predict blueprint segmentation during the pretraining, as advised in [2].
- NoAux-PAST: We eliminate the auxiliary pose regression task, focusing solely on predicting the assembly sequence using the same network.

C. Experimental Results

Table I, Fig. 4, and Fig. 5 summarize the quantitative and qualitative results, which are elaborated upon in this section. For more results, we refer readers to our video.

Multi-level assembly planning. As outlined in Sections I and III, our methodology breaks down assembly planning into three distinct phases. This approach achieved assembly planning in 1565.9 ms, with an 82.9% success rate across our 500 multi-part assemblies test set. Figure 4 illustrates an example of assembling parts from scratch. Specifically, part movement planning averaged 1476.9 ms with a 100% success rate, while contact point optimization took 36.8 ms

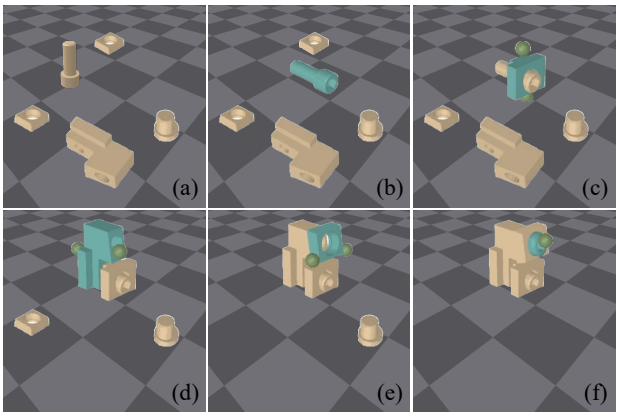

Fig. 4: Assembly planning results (a-f). In each step, the part colored blue indicates the one in motion, while the yellow parts signify those that are stationary. In (c, d, e), the dual green spheres denote feasible grasp points. In (f), a solitary green sphere highlights the designated pushing point.

with 100% success rate using multi-threaded computation and the CVXPY solver [40]. Unlike our method, RL-based assembly planning [11] exhibits inferior performance when evaluated in our setting, achieving a 63.9% assembly success rate. While prior works showed promise with simpler geometries [10], [11], we posit that end-to-end policies may struggle to handle complex geometries and assembly reasoning simultaneously.

Generalization to novel assemblies. From Table I, we see that PAST consistently outperforms other neural sequence inference models, such as NSM, DGL, Seg-PAST, and NoAux-PAST in both one-step and full-sequence prediction tasks. Unlike NSM, which employs a discriminator for target shape understanding and fails to capture the assembly geometries' distribution, PAST leverages the target assembly blueprint as input and can extract direct features from the target assembly. DGL, which represents parts as a graph and updates features at the node level, struggles to model geometries from other parts and the target shape. In contrast, PAST aggregates features at the point level, thus enhancing geometric understanding, consequently yielding superior learning outcomes.

Further, from Table I, we also see from the ablated PAST variants that incorporating auxiliary tasks, such as pose regression or part segmentation, significantly improves network performance. This improvement is attributed to the additional guidance these tasks offer, enhancing the network's understanding of part interactions, which are key for assembly sequence inference. Interestingly, target segmentation underperforms compared to pose regression, possibly because some points in the target assembly, like those corresponding to assembled parts, are not supervised during training. Further, as is expected, ATA takes significant computing as it uses enumeration and simulation for assembly sequences. Instead, PAST, which shows promising accuracy, has a dramatically short computing time, making it well-suited for real-world robotic assembly. In Figure 5, we analyze the impact of the number of parts on sequence inference accuracy and auxiliary pose regression error. The

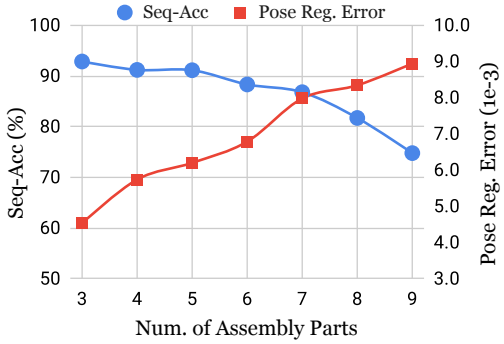

Fig. 5: Sensitivity analysis for sequence inference accuracy and auxiliary pose regression error.

results indicate that increased part count leads to reduced sequence accuracy and higher pose error, corroborating results from [10]: the complexity of the assembly problem increases with the number of parts in the target blueprint.

Disassembly Planning. As noted earlier, while our D4PAS construction bears similarities to [13], we enhance disassembly planning through: (i) employing GPU-accelerated simulation and (ii) applying part-centric forces. Enabling parallel computations thereby reducing compute time for disassembly planning to 3592.0 ms against 24312.6 ms. This efficiency not only allows for the enumeration of feasible assembly sequences within a manageable timeframe but also establishes a performance benchmark for future research in end-to-end robotic assembly learning [10]. Second, we apply part-centric disassembly forces aligned with the parts' inertia axes, as supported by findings in [41], [42]. Although this adjustment in the force application coordinate system may seem minor, it led to a notable increase in planning success rate: 92.7% in our approach versus 83.8% in [13].

VI. Conclusion

This paper makes several key contributions to robotic assembly. First, we introduce a multi-level framework for generating assembly plans, encompassing part sequences, motions, and contact points. Second, we unveil the Part Assembly Sequence Transformer (PAST) for inferring feasible assembly sequences based on target blueprints and part geometries. Third, we offer a large-scale benchmark dataset for part assembly sequence (D4PAS) featuring thousands of physically validated sequences. Post-sequence inference, we employ motion planning and contact optimization to complete part assembly. Our evaluations show that PAST and the overall algorithm match previous simulation-based methods but with significantly reduced computation time. Additional results and visualizations can be found in the supplementary videos.

The present work has some limitations. Our approach assumes one moving part at a time, as well as ideal part sensing and localization, which may limit practical robotic execution. Further, additional mechanisms may be needed to hold the assembly. Future work will focus on fixing these limitations while enabling real-world robotic assembly and tool use.

REFERENCES

[1] S. Dorn, N. Wolpert, and E. Schömer, "An assembly sequence planning framework for complex data using general voronoi diagram," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9896–9902.

[2] Y. Li, A. Zeng, and S. Song, "Rearrangement planning for general part assembly," in *Conference on Robot Learning*. PMLR, 2023.

[3] Y. Li, K. Mo, L. Shao, M. Sung, and L. Guibas, "Learning 3d part assembly from a single image," *European conference on computer vision*, 2020.

[4] J. Huang, G. Zhan, Q. Fan, K. Mo, L. Shao, B. Chen, L. Guibas, and H. Dong, "Generative 3d part assembly via dynamic graph learning," in *Advances in Neural Information Processing Systems*, 2020.

[5] Y.-C. Chen, H. Li, D. Turpin, A. Jacobson, and A. Garg, "Neural shape mating: Self-supervised object assembly with adversarial shape priors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory: Fast contact for robotic assembly," in *Robotics: Science and Systems*, 2022.

[7] S. Jin, X. Zhu, C. Wang, and M. Tomizuka, "Contact pose identification for peg-in-hole assembly under uncertainties," in *American Control Conference (ACC)*, 2021, pp. 48–53.

[8] X. Zhang, S. Jin, C. Wang, X. Zhu, and M. Tomizuka, "Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9881–9887.

[9] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-rl for insertion: Generalization to objects of unknown geometry," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6437–6443.

[10] S. K. S. Ghasemipour, D. Freeman, B. David, S. S. Gu, S. Kataoka, and I. Mordatch, "Blocks assemble! learning to assemble with large-scale structured reinforcement learning," in *International Conference on Machine Learning*, 2022.

[11] M. Yu, L. Shao, Z. Chen, T. Wu, Q. Fan, K. Mo, and H. Dong, "Roboassembly: Learning generalizable furniture assembly policy in a novel multi-robot contact-rich simulation environment," in *International Conference on Robotics and Automation (ICRA)*, 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, and W. Matusik, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Trans. Graph.*, vol. 41, no. 6, 2022.

[14] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation.*, vol. 2, 2000, pp. 995–1001 vol.2.

[15] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine, "Offline meta-reinforcement learning for industrial insertion," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6386–6393.

[16] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.

[17] S. Dasari, A. Gupta, and V. Kumar, "Learning dexterous manipulation from exemplar object trajectories and pre-grasps," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3889–3896.

[18] X. Zhu, W. Lian, B. Yuan, C. D. Freeman, and M. Tomizuka, "Allowing safe contact in robotic goal-reaching: Planning and tracking in operational and null spaces," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8120–8126.

[19] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[20] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation," *arXiv preprint arXiv:2305.12821*, 2023.

[21] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology*, vol. 59, pp. 335–349, 2012.

[22] Q. Su, "A hierarchical approach on assembly sequence planning and optimal sequences analyzing," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 1, pp. 224–234, 2009.

[23] L. H. De Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," in *1989 IEEE International Conference on Robotics and Automation*, 1989.

[24] S. Ghandi and E. Masehian, "Review and taxonomies of assembly and disassembly path planning problems and approaches," *Computer-Aided Design*, vol. 67, pp. 58–86, 2015.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[26] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3524–3531.

[27] X. Zhu, J. Ke, Z. Xu, Z. Sun, B. Bai, J. Lv, Q. Liu, Y. Zeng, Q. Ye, C. Lu, M. Tomizuka, and L. Shao, "Diff-lfd: Contact-aware model-based learning from visual demonstration for robotic manipulation via differentiable physics-based simulation and rendering," in *Conference on Robot Learning*. PMLR, 2023.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[30] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[31] C.-F. R. Chen, Q. Fan, and R. Panda, "CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification," in *International Conference on Computer Vision (ICCV)*, 2021.

[32] X. Zhu, Y. Zhou, Y. Fan, L. Sun, J. Chen, and M. Tomizuka, "Learn to grasp with less supervision: A data-efficient maximum likelihood grasp sampling loss," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 721–727.

[33] X. Zhu, L. Sun, Y. Fan, and M. Tomizuka, "6-dof contrastive grasp proposal network," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6371–6377.

[34] Y. Fan, X. Zhu, and M. Tomizuka, "Optimization model for planning precision grasps with multi-fingered hands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 1548–1554.

[35] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[36] S. Boyd and B. Wegbreit, "Fast computation of optimal contact forces," *Robotics, IEEE Transactions on*, vol. 23, 01 2008.

[37] J. Huang, G. Zhan, Q. Fan, K. Mo, L. Shao, B. Chen, L. Guibas, and H. Dong, "Generative 3D part assembly via dynamic graph learning," in *Advances in Neural Information Processing Systems*, 2020.

[38] L. Ma, J. Gong, H. Xu, H. Chen, H. Zhao, W. Huang, and G. Zhou, "Planning assembly sequence with graph transformer," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 395–12 401.

[39] P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *arXiv*, 2018.

[40] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[41] K. D. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, "Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences," *ACM Transactions on Graphics*, vol. 40, no. 4, 2021.

[42] K. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, "Joinable: Learning bottom-up assembly of parametric cad joints," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 15 849–15 860.