Fabrica: Dual-Arm Assembly of General Multi-Part Objects via Integrated Planning and Learning

Yunsheng Tian[1,†], **Joshua Jacob**[1], **Yijiang Huang**[2], **Jialiang Zhao**[1], **Edward Gu**[1], **Pingchuan Ma**[1], **Annan Zhang**[1], **Farhad Javid**[3], **Branden Romero**[1], **Sachin Chitta**[3], **Shinjiro Sueda**[4], **Hui Li**[3,†], **Wojciech Matusik**[1,†]
[1]MIT CSAIL, [2]ETH Zurich, [3]Autodesk Research, [4]Texas A&M University

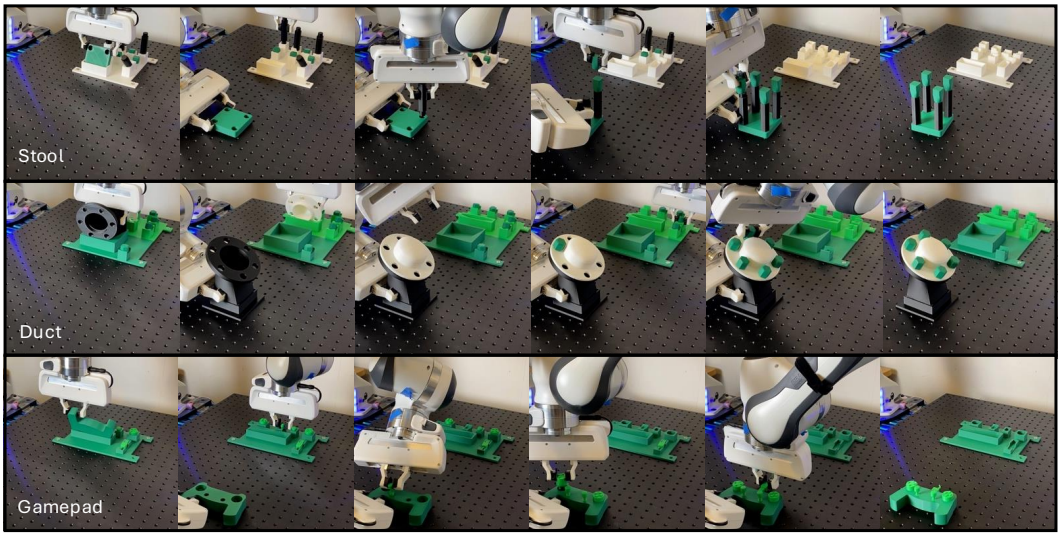


Fig. 1: Our proposed dual-arm robotic system demonstrates adaptive manipulation and assembly capabilities for diverse multi-part objects. The system combines offline task-oriented planning and optimization to address sequencing, grasping, and motion planning for long-horizon assembly tasks. For robust online control, it utilizes guidance from the offline plan to learn assembly skills that generalize effectively across diverse object geometries, assembly paths, and grasp poses.

Abstract: Multi-part assembly poses significant challenges for robots to execute long-horizon, contact-rich manipulation with generalization across complex geometries. We present Fabrica, a dual-arm robotic system capable of end-to-end planning and control for autonomous assembly of general multi-part objects. For planning over long horizons, we develop hierarchies of precedence, sequence, grasp, and motion planning with automated fixture generation, enabling general multi-step assembly on any dual-arm robots. The planner is made efficient through a parallelizable design and is optimized for downstream control stability. For contact-rich assembly steps, we propose a lightweight reinforcement learning framework that trains generalist policies across object geometries, assembly directions, and grasp poses, guided by equivariance and residual actions obtained from the plan. These policies transfer zero-shot to the real world and achieve 80% successful steps. For systematic evaluation, we propose a benchmark suite of multi-part assemblies resembling industrial and daily objects across diverse categories and geometries. By integrating efficient global planning and robust local control, we showcase the first system to achieve complete and generalizable real-world multi-part assembly without domain knowledge or human demonstrations. Project website: fabrica.csail.mit.edu

Keywords: Assembly, Planning, Reinforcement Learning, Benchmark

[†]Correspondence to: yunsheng@mit.edu, hui.xylo.li@autodesk.com, and wojciech@mit.edu

1 Introduction

Multi-part assemblies are prevalent in home and industrial settings. Robotic assembly of multi-part objects presents a longstanding challenge: long-term planning to map CAD models to robot programs and robust control skills to achieve high precision and adaptivity during contact-rich interactions. However, most assembly robots today are programmed manually with specially designed infrastructures, and the program is executed repetitively using a stiff controller. As a result, they take substantial time to adapt to new production demands and are highly sensitive to uncertainties.

Despite recent progress in sim-to-real transfer of contact-rich part insertion skills [1, 2, 3], current robotic systems are still not capable of assembling general multi-part objects. Prior research has primarily focused on two-part, top-down insertion using a single robot arm, but multi-part assembly requires diverse insertion and grasping poses and a bi-manual operation that frequently changes which part to hold to counter-balance the insertion force from the other hand. This presents new challenges to planning and control. First, jointly finding an assembly-hold sequence, physically stable grasps, and collision-free robot motion presents a hybrid (discrete-continuous) optimization problem in a large search space. Second, control policies for part insertion must be robust to misalignment and uncertainty, while being able to generalize across a wide range of part geometries.

We tackle these challenges by building a general planning and control system for flexible, dual-arm assembly of multi-part objects, with zero-shot sim-to-real transfer. Our contributions include:

Algorithms: We propose a hierarchical dual-arm planner to plan and optimize the assembly-hold sequence, grasps, and robot motion. For contact-rich steps, we learn generalist reinforcement learning (RL) policies utilizing equivariant representations guided by planned motion to achieve robustness.

Systems: We build a real-world system that can map a CAD assembly model to robot execution that alternates between tracking planned motions and reactive control policies. To our knowledge, this is the first system that autonomously achieves all phases of a multi-part assembly problem: from automatic pickup fixture design, to sequence, grasps, and motion planning, to insertion. Our system is tested on commonly used robotics hardware and can be generalized to different dual-arm robots.

Benchmarks: We design a benchmark suite of 7 multi-part assemblies ranging from 5 to 9 parts, and our system can assemble them robustly in both simulation and real-world system.

2 Related Work

Prior work on multi-part assembly is heavily focused on planning assembly sequences and paths, including geometric reasoning [4], sampling-based motion planners [5, 6, 7], and RL for combinatorial sequence search [8, 9]. Recently, physics-based motion planning [10] has shown success in assembling many complex parts with tight clearances. In addition, realistic kinematic and dynamic constraints have been considered in sequence planning for real-world robot setups [11, 12, 13]. However, planning alone struggles with execution uncertainties, and stability- or efficiency-optimal plans remain underexplored. While robust and efficient robotic systems have been built for tasks like assembling IKEA chairs [14], LEGO blocks [15], and structural elements [16, 17, 18, 19, 20, 21, 22], these are domain-specific and lack generalizability. In contrast, our planner generalizes across diverse multi-part assemblies, employs hierarchical structure and parallelization for efficiency, and explicitly optimizes stability to enhance downstream control robustness.

Even with given assembly plans, executing contact-rich assembly remains challenging due to tight clearances, system uncertainties, and the need for generalization. RL has shown promise in addressing these issues, combining motion planning with policy learning from CAD models or supervised trajectories [23, 24], and leveraging accurate simulations for motion generation and policy training [11, 25]. Sim-to-real transfer [26, 3, 27, 2] and real-world RL [28] have enabled high-precision insertion, while some efforts [29] explore multi-step tasks. However, they all primarily work under simplified settings like top-down insertions and fixed grasps, which are insufficient for multi-part assembly where side-way insertions or tilted grasps are necessary. Imitation learning ap-

proaches [30, 31] support complex multi-step skill learning but lack robustness and generality. While spatial-equivariant techniques have improved generalization in other domains [32, 33, 34, 35, 36], they remain underexplored for assembly. Notably, Seo et al. [37] learn an SE(3)-equivariant gain scheduling policy, but without varying grasps or geometries. Existing benchmarks focus on narrow tasks [38, 39, 40, 41, 42], limiting the evaluation of generalization. In contrast, we demonstrate that combining planning with equivariant generalist policies, for the first time, enables multi-part assembly over diverse geometries, paths, and grasps without any human demonstration.

3 Planning Multi-Step Dual-Arm Assembly

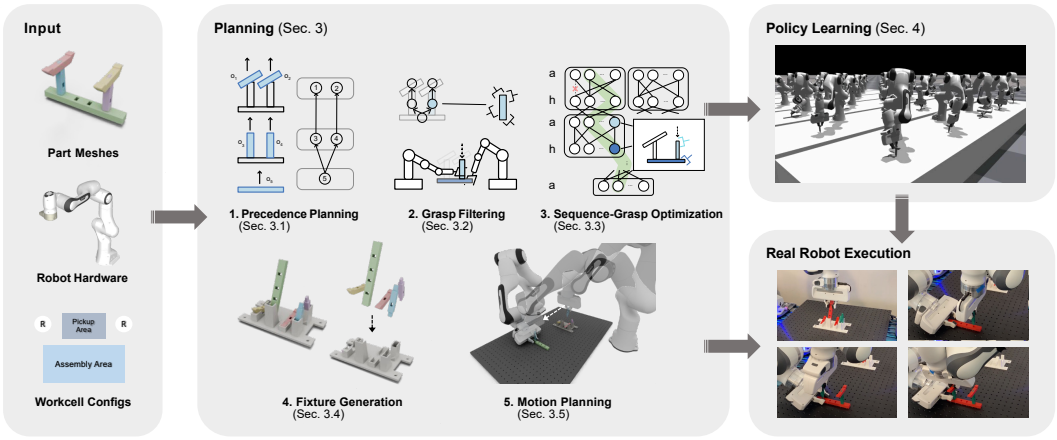


Figure 2: System overview. Fabrica takes part meshes and hardware configurations as inputs. It plans sequences, grasps, fixture designs, and motions through a multi-stage planner, and learns RL policies for all insertion steps, which are deployed together on real robots to complete the assembly.

Given a n-part assembly with parts indexed by $o \in O$, we compute a plan to manipulate all parts from the initial poses p_O^0 to the goal poses $p_O^G \in SE(3)$ under all physical constraints. We focus on sequential, collaborative manipulation that alternates between robot R_a assembling a part and another robot R_h holding a part to stabilize the sub-assembly. Then, we train control policies for precise contact-rich assembly steps. Finally, our system execution alternates between open-loop planned motions and closed-loop reactive policies. Fig. 2 provides an overview of the system.

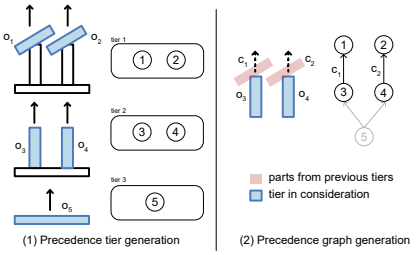
We formulate planning as optimizing assembly-hold sequences ϕ, grasps σ, and robot motions π:

$$\min_{\phi,\sigma,\pi} E\big(\Phi_{i=1}^{n} \vec{f}(\phi_{1:i}, \sigma_{i-1:i}, \pi_i)\big) \quad \text{s.t.} \quad C_{\text{prec}}(\phi) \le 0,\ C_{\text{kin}}(\phi,\sigma,\pi) = 0,\ C_{\text{col}}(\phi,\sigma,\pi) \le 0 \quad (1)$$

The sequence $\phi = [o_{a,1}, o_{h,2}, o_{a,2}, \cdots, o_{h,n}, o_{a,n}]$ is an ordering of parts to be held (h) and assembled (a), with $\sigma = [g_{a,1}, g_{h,2}, g_{a,2}, \cdots, g_{h,n}, g_{a,n}]$ including grasp $g \in SE(3)$ for each step. The robot motion π is divided based on the mode families [43, 44] and skills: $\pi = [\underbrace{\tau^f_{a,1}, \tau^g_{a,1}, \tau^a_{a,1}}_{\pi[a,1]}, \underbrace{\tau^f_{h,2}}_{\pi[h,2]}, \underbrace{\tau^f_{a,2}, \tau^g_{a,2}, \tau^a_{a,2}}_{\pi[a,2]}, \cdots]$ where each assembly task $\pi[a,i]$ for R_a contains (1) a transit motion $\tau^f_{a,i}$ with its hand free, (2) a transfer motion $\tau^g_{a,i}$ grasping an object, and (3) an assembly motion $\tau^a_{a,i}$ for part insertion. A hold task $\pi[h,i]$ only involves a hold transfer motion $\tau^f_{h,i}$. The cost function composes step-wise objective vectors $\vec{f}$ that evaluate the quality of each step, $\Phi : \mathbb{R}^{|\vec{f}|\times n} \to \mathbb{R}^{|\vec{f}|}$ aggregates objectives across steps (e.g., sum or max), and E maps the result to a scalar (e.g., weighted sum). $C_{\text{prec}}, C_{\text{kin}}, C_{\text{col}}$ represent part precedence, kinematic, and collision constraints respectively. Please refer to App. B for detailed constraint formulation.

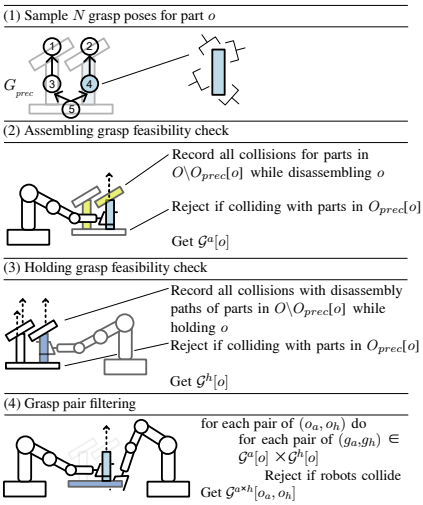
Solving for feasible or even optimal solutions in a joint manner is intractable. We present a hierarchical approach to decompose it into simpler subproblems for efficient computation with optimality guarantees under assumptions (A1)-(A5). The complete pesudocode can be found in App. C.

3.1 Part Precedence Planning



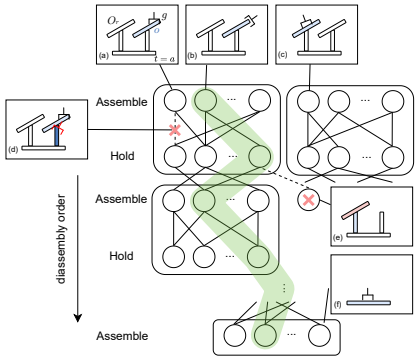
To evaluate constraint C_{prec}, we propose an algorithm to determine the complete precedence relationships for assembling all parts O. First, we define precedence tier as a group of parts that can be removed independently of one another. Tiers are ordered so that parts in earlier tiers must be disassembled before those in later ones. To iteratively construct all tiers, we use a physics-based motion planner [10] to find all parts that can be disassembled without interfering with the rest, which are grouped into the current tier. We then remove these parts and repeat the process on the remaining assembly until each part is assigned to a tier. Next, we build a precedence graph G_{prec} that encodes the minimal set of ordering constraints that any collision-free assembly sequence must follow. Each node in G_{prec} is a part, and a directed edge $o_i \rightarrow o_j$ means that o_i must be assembled before o_j because o_j blocks the (dis)assembly path τ_{o_i} planned during tier generation. For each part o, we define its precedence set $O_{\text{prec}}[o] = \{o' | (o' \rightarrow o) \in G_{\text{prec}}\}$ as all parts that must be assembled before it.

3.2 Dual-Arm Grasp Filtering



We aim to identify valid grasp pairs $\mathcal{G}^{a \times h}[o_a, o_h]$ for each assembly-hold part pair (o_a, o_h) which support insertion and holding without colliding with preceding parts of o_a and o_h. Since searching the full 6-DoF space is infeasible, we assume feasible grasps exist in a dense, finite set of grasps, see (A5). Because each grasp must be checked for collisions with the current subassembly $\psi_{1:i}$, doing this online results in repeated and expensive checks against many part combinations. To speed up, we precompute valid grasps offline by sampling grasp candidates and performing parallelized inverse kinematics (IK) and collision checks for both arms. In practice, we simulate the robot following the part's disassembly path τ_o and reject it if the motion collides with the precedence set $O_{\text{prec}}[o]$. All collisions and motions are recorded for reuse in later stages. Similarly, we check if a grasp can securely hold a part while disassembling other parts without collision. Finally, for each part pair (o_a, o_h), we filter feasible grasps (g_a, g_h) by checking interarm collisions under computed IK.

3.3 Dual-Arm Sequence-Grasp Optimization



With all valid grasps computed, we now solve for the optimal sequence ϕ^* and grasps σ^* in Eq. 2. We formulate this as a state-space search problem and construct a directed state tree T_G, where each node represents a partial assembly state $s = (t, O_r, o, g)$ consisting of robot task t (assemble or hold), assembled parts O_r, part being grasped o, and grasp pose g. Starting from root nodes (complete assembly), we recursively expand the tree by alternating between assembling and holding, pruning states that violate constraints. Valid transitions must also respect precomputed grasp feasibility between successive steps. All collision and motion feasibility checks are reused from the earlier filtering stage. Each transition is scored by a grasp stability vector $\vec{f}$, capturing objectives such as supportiveness of the held part, frequency of grasp switches, torque stability, and contact area, which are designed to be lightweight yet effective for downstream control. We apply dynamic programming (DP) to propagate the best cumulative scores through the tree and identify the optimal solution. Please find more details in App. C.3.

3.4 Grasp-Aware Pickup Fixture Generation

For precise pickup, we develop a software-hardware co-design approach to automatically generate a fixture that stabilizes and orients each part for top-down pickup, based on planned grasps in Sec. 3.3. This removes the need for reorientation or regrasping between pickup and assembly, allowing the system to focus on the core assembly challenges. We first determine each part's pickup pose in the world frame, with its orientation defined by the rotation from the assembly grasp to a top-down grasp. We then compute pickup positions by packing parts on the XY plane to avoid collisions between parts and the gripper. To reduce material usage and workspace area, we model this as a bin-packing problem and solve it with an iterative algorithm that alternates between packing, collision checking, and resolution. Finally, we generate the fixture by creating mold cavities based on the part geometries and poses, ensuring stable placement. See details in App. C.4 and examples in Fig. 3.

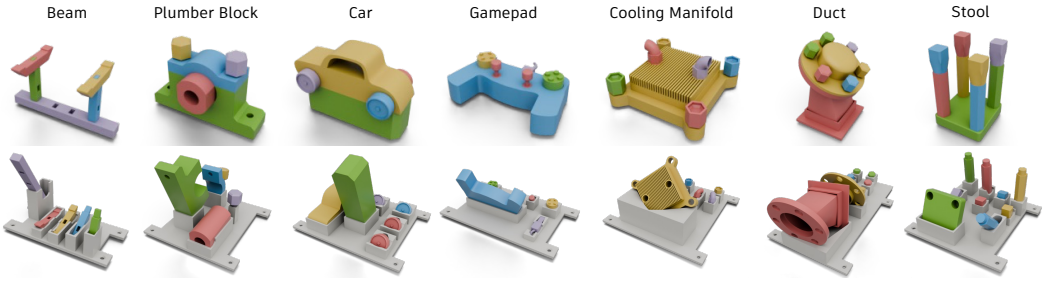


Figure 3: Top: benchmark assemblies. Bottom: the auto-generated pickup fixtures in Sec. 3.4.

3.5 Motion Planning for Transit and Transfer

Finally, RRT-Connect [45] plans for all remaining transit and transfer motions, i.e., $\tau^f_{a,i}, \tau^g_{a,i}, \tau^f_{h,i}$, which can be parallelized since all start and goal states of motions are provided from earlier stages.

4 Learning General Single-Step Assembly Policy

Once the full assembly plan is computed, the next challenge is to track it reliably in the real world. We use a hybrid controller that alternates between tracking the pre-planned transit and transfer motions, and an RL-based reactive controller for contact-rich assembly steps. The controller must generalize across variations in object geometry, grasp poses, and assembly directions. To this end, we design a lightweight yet highly effective RL framework for training a generalist assembly policy. Given a pre-planned insertion path τ_o, our goal is to guide the part from its noisy initial pose $\hat{\tau}_o[0]$ to the goal pose $\hat{\tau}_o[1]$, accounting for uncertainty in grasp and mating part geometry. We frame this as a Markov decision process (MDP) and learn a policy $\pi : \mathcal{O} \to \mathbb{P}(\mathcal{A})$ that maximizes expected rewards over time $\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r(s_t)]$. We use proximal policy optimization (PPO) [46] to train a stochastic policy in simulation, which is then transferred to the real world without additional fine-tuning.

4.1 Path-Centric Coordinate Transformation

Humans naturally reuse the same assembly skills across different objects, regardless of their poses or motions. We emulate this ability by designing a problem space transformation [47] with SE(3)-equivariance, which maps all straight-line assembly motions in the world frame into equivalent top-down insertions in the task frame, allowing the RL agent to perceive them in a unified way.

Formally speaking, given an assembly path τ with a pre-assembly position p_d and the assembled position p_a, we define a path-centric transformation $\mathbf{T}$ conditioned on τ such that $\mathbf{T}(p_d) = (0, 0, 0)$ and $\mathbf{T}(p_d) - \mathbf{T}(p_a) = (0, 0, \|p_d - p_a\|)$. Thus, the agent's observation is the transformed position of the part being assembled (under unknown noise), and its action is the transformed delta position, i.e., the ideal position change in the path-centric frame. Thanks to equivariance, the agent only needs to learn top-down insertions and can omit orientation from both observations and actions,

simplifying the learning setup. We use a task-space impedance (TSI) controller to enable smooth and compliant insertions, with impedance gains similarly transformed into the path-centric frame to maintain consistent behavior across different assembly directions. This design ensures that the observation and action spaces are minimal yet essential, facilitating generalizability among different assembly tasks, and is transferable to different robot arms and end effectors.

4.2 Plan-Guided Residual Action

We find that guidance from the planned open-loop action helps learning by injecting prior knowledge about the coarse assembly direction. Thus, we adopt the idea of residual action [48, 31] in RL, where the policy outputs only the corrective action on top of the open-loop action, allowing the policy to focus on refining the assembly rather than learning the full assembly behavior. In practice, residual action warm-starts policy learning and typically leads to faster and better convergence.

4.3 Minimalist Reward Design and Sim-to-Real Transfer

Surprisingly, our insertion reward is simply the negative L2 distance from the current part position to the goal position. This form is dense and sufficient enough for learning effective local insertion policies as our initial state is the pre-assembled pose given by the planner, which is already in the proximity of the goal thus does not require expensive exploration or complex reward engineering.

Due to the sim-to-real gap from misaligned dynamics, we adopt 1) domain randomization with 3mm-noised initialization on object pose during training and 2) Policy-Level Action Integrator (PLAI) [26] during deployment to ease the sim-to-real transfer of RL policies, which improves action consistency by incrementally applying policy outputs to the last desired state instead of the current state. PLAI applies policy actions as $s^d_{t+1} = s^d_t \oplus \Pi(o_t)$, where s^d_t represents the desired state at time t, $\Pi(o_t)$ is the policy action computed based on the current observation o_t, and $\oplus$ denotes the composition operation, instead of the nominal approach $s^d_{t+1} = s_t \oplus \Pi(o_t)$ which is prone to error accumulation.

5 Experiments

5.1 Benchmark Suite and Experimental Setup

We develop a diverse benchmark suite spanning furniture, toys, and industrial equipment, which includes **beam** (5 parts), **plumber block** (5 parts), **car** (6 parts), **gamepad** (6 parts), **cooling manifold** (7 parts), **duct** (8 parts), and **stool** (9 parts). These assemblies cover various geometries and connection types found in real-world applications, with both top-down and sideway insertions, and are feasible for dual-arm robots with parallel grippers. For planning in simulation, we demonstrate on several different robots, including Franka Emika Panda, UFactory xArm7, and UR5e with different grippers. We use Panda for systematic evaluations of policy training and real-world execution. See more details on the experimental setup and hyper-parameters in App. D.

5.2 Planning Multi-Step Assembly in Simulation

Efficiency: Table 1 shows the breakdown of planning time by stages for different assemblies. Our overall speed is on the order of minutes to solve for optimal plans given efficient parallelization.

Optimality: Table 2 shows the objective scores of optimized sequences surpassing the random ones by a large margin with priorities from f_1 to f_4 (see App. C.3 for details on the score definitions).

Generality: Please see App. A for visual demonstrations of planning with different robot arms (Panda, xArm7, UR5e) and grippers. Our planning framework is general to any given hardware.

5.3 Learning Single-Step Assembly in Simulation

We use Isaac Gym [49] for training RL policies and performing simulation evaluations, with the PPO code from RL Games [50]. Table 3 presents the average % of successful steps for assembling our

Table 1: Planning runtime breakdown of each assembly. Stages marked with * are parallelized, while others have yet to be parallelized.

Assembly	Runtime (s)					
	Prec*	Grasp*	Seq	Fixture	Motion	**Total**
Beam	19.7	35.7	0.2	1.7	115.9	173.2
Plumber	21.6	31.8	12.6	0.9	118.9	185.7
Car	23.4	52.9	0.9	1.9	127.4	206.4
Gamepad	22.2	37.2	4.5	3.6	117.4	184.8
Manifold	20.9	162.7	5.2	1.9	149.6	340.4
Duct	93.9	102.3	208.1	2.5	185.9	592.7
Stool	57.2	109.1	8.0	4.0	324.5	502.7

Table 2: Objective comparisons between optimal and random sequences. Higher is better for f_1, f_4; lower is better for f_2, f_3.

Assembly	Objective Values (Optimal / Random)			
	$f_1 \uparrow$	$f_2 \downarrow$	$f_3 \downarrow$	$f_4 \uparrow$
Beam	4.00 / 0.88	4.00 / 0.88	0.48 / 0.39	119.8 / 6.8
Plumber	4.00 / 1.27	2.00 / 1.79	0.18 / 0.33	293.0 / 93.4
Car	4.00 / 1.72	1.00 / 1.64	0.21 / 0.94	140.6 / 35.2
Gamepad	5.00 / 1.76	1.00 / 1.56	1.40 / 0.63	428.6 / 52.1
Manifold	6.00 / 2.62	1.00 / 2.17	0.18 / 0.69	12.5 / 18.7
Duct	6.00 / 3.00	1.00 / 2.88	0.09 / 0.39	536.0 / 101.3
Stool	8.00 / 3.43	6.00 / 3.24	0.03 / 0.54	322.2 / 87.9

Table 3: % of successful steps without intervention in simulation evaluations.

Method	% of Successful Steps without Intervention (Simulation)						
	Beam	Plumber Block	Car	Gamepad	Cooling Manifold	Duct	Stool
Open-Loop Tracking	21.48	24.22	2.34	2.34	3.91	18.75	0.00
Part Specialist Policy (PS)	**98.63**	84.08	**90.82**	**87.60**	**94.63**	**100.00**	**78.91**
Assembly Specialist Policy (AS)	**99.12**	**97.46**	70.12	**88.87**	**95.02**	96.58	76.66
Assembly Generalist Policy (AG)	**98.83**	81.64	60.55	71.48	89.06	89.84	58.59

benchmark assemblies in simulation across 1024 random trials using different methods: 1) **Open-Loop Tracking**: A baseline that strictly follows the pre-planned path without feedback correction. 2) **Part Specialist Policy (PS)**: Policies trained on individual pairs of parts. 3) **Assembly Specialist Policy (AS)**: Policies trained on all parts within a single assembly. 4) **Assembly Generalist Policy (AG)**: Policies trained on all parts from all assemblies in our suite, aiming for broad generalization.

The results show that open-loop tracking exhibits the lowest success rates across all assemblies, indicating its limitations in handling uncertainties and variations. The AS policy demonstrates competitive performance as the PS policy, suggesting that a shared policy across different parts in an assembly can generalize well. It may sacrifice some part-specific optimization, but can transfer the knowledge between similar parts. The AG policy, while slightly less effective than the other RL counterparts, still demonstrates robust performance, suggesting that learning a single shared policy across different assemblies is promising, given the equivariant representations. Furthermore, the success rates vary across different assemblies, with simpler assemblies like the Beam and Cooling Manifold achieving higher performance across all methods, while more complex assemblies such as the Gamepad and Stool exhibit lower success rates due to their intricate geometries and constraints.

5.4 Executing Multi-Step Assembly in Real World

Table 4 shows the % of successful steps on benchmark assemblies evaluated in the real world (step-wise statistics), and Table 5 shows the multi-step cumulative success rates with 0/1/2 total interventions for failure recovery (overall statistics). All numbers are averaged across three complete multi-step real experiments, which translate to thousands of total assembly steps. We deploy stochastic policies with state-based success detection and allow up to three trials per step until success. For qualitative results on real-world multi-step executions, please refer to Fig. 1 and App. A.

Ours: We use both AS and AG policies for real-world comparisons. For AG, we perform out-of-distribution (OOD) evaluations by training 7 generalist policies, where each one is trained on the other 6 benchmark assemblies (excluding the test assembly). Remarkably, these OOD generalist policies still achieved comparable performance to specialist policies trained directly on the test assembly, which indicates that through our approach, insertion strategies learned from a diverse set of assemblies can effectively transfer to novel, unseen assemblies.

Baseline: Since Fabrica is the first to assemble general multi-part objects with only CAD input, identifying a comparable SOTA baseline is challenging. The closest work is ASAP [11], which performs single-arm kinematic feasibility search without sequence/grasp optimization or closed-loop control. To compare, we adapted it by planning with dual-arm and adding our RL policy. Without optimized part sequencing and grasping, ASAP performed substantially worse, often struggling even with two interventions, which emphasizes the contributions of our planning optimizations.

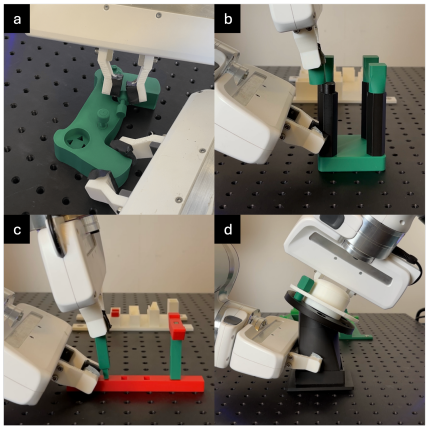
Table 4: % of successful steps without intervention in real-world evaluations.

Method		% of Successful Steps without Intervention (Real World)							
		Beam	Plumber Block	Car	Gamepad	Cooling Manifold	Duct	Stool	**Overall**
Ours	AS	**75**	**83**	80	**87**	72	71	**92**	**80**
	AG (OOD)	67	75	**93**	80	72	**81**	**92**	**80**
Baseline	ASAP (Adapted)	50	42	67	33	55	52	75	55
	Open-Loop Tracking	42	25	20	20	17	14	21	23
Ablation	w/o Part Seq Optim	**75**	75	73	40	61	71	**92**	71
	w/o Grasp Optim	42	**83**	47	60	67	67	75	64
	w/o Path-Centric Transform	67	67	20	53	**78**	38	**92**	61

Table 5: Multi-step cumulative success rate with 0/1/2 interventions in real-world evaluations.

Method		Multi-Step Cumulative Success Rate with 0/1/2 Interventions (%) (Real World)							
		Beam	Plumber Block	Car	Gamepad	Cooling Manifold	Duct	Stool	**Overall**
Ours	AS	0/100/100	33/100/100	0/100/100	33/100/100	0/ 67/ 67	0/ 0/100	33/100/100	**15/ 81/** 95
	AG (OOD)	0/ 67/100	0/100/100	67/100/100	0/100/100	0/ 33/100	0/ 67/100	33/100/100	10/ **81/100**
Baseline	ASAP (Adapted)	0/ 0/100	0/ 0/ 67	0/ 33/100	0/ 0/ 0	0/ 0/ 33	0/ 0/ 0	0/ 0/100	0/ 5/ 57
	Open-Loop Tracking	0/ 0/ 67	0/ 0/ 33	0/ 0/ 0	0/ 0/ 0	0/ 0/ 0	0/ 0/ 0	0/ 0/ 0	0/ 0/ 14
Ablation	w/o Part Seq Optim	0/100/100	33/ 67/100	0/ 67/100	0/ 0/ 0	0/ 0/ 67	0/ 0/100	33/100/100	10/ 48/ 81
	w/o Grasp Optim	0/ 0/ 67	33/100/100	0/ 0/ 33	0/ 0/100	0/ 0/100	0/ 0/ 67	0/ 0/100	5/ 14/ 81
	w/o Path-Centric Transform	0/ 67/100	0/ 67/100	0/ 0/ 0	0/ 0/ 67	0/ 67/100	0/ 0/ 0	33/100/100	5/ 43/ 67

Ablation: To understand how much optimizing part sequences and grasps helps, we conduct ablation studies on our method by removing those optimizations respectively. We observed that suboptimal sequencing often caused instability due to inadequate support of critical neighboring parts and more part drifts due to unnecessary re-grasps. Meanwhile, suboptimal grasp selection frequently caused part slippage due to insufficient contact area or inadequate resistance to external torques. Thus, our planner inherently accounts for control-level uncertainties, and results demonstrate that selecting effective part sequences and grasps significantly enhances assembly reliability. For control, we observed that our path-centric transformation is crucial for generalizing across varying assembly directions. Policies trained without it perform significantly worse when multiple directions are involved. For more studies on the effects of path-centric transformation and residual actions introduced in Sec 4, and success increase w.r.t. the number of trials per step, please see App. E.



Failures: We observed a noticeable gap between simulation and real-world performances. Thus, we present a detailed analysis of common failure cases shown in the right figure: a) Small parts slip between gripper pads during insertion attempts; b) Cumulative error accrued during the assembly of large assemblies increases the displacement error of final part insertions; c) The holding gripper is not modeled during RL training, causing unexpected part obstructions in the real world; d) Unstabilized parts shift previously assembled parts during insertion. We assume a 3mm noise in simulation given that the base part is stably held. However, many sources of real-world error lead to much more significant errors than simulated, which are non-trivial challenges for future work. Due to these failures, all methods achieve near-zero multi-step success rates without intervention due to inherently challenging steps causing consistent failures. However, with minimal interventions, our method significantly outperforms others, reaching 81% success with one intervention and 95% with two.

6 Conclusion

We presented Fabrica, a dual-arm robotic system that innovates and integrates global hierarchical planning with local generalist policy learning for autonomous multi-part assembly. To support reproducible and rigorous evaluation, we introduced a comprehensive benchmark suite covering diverse multi-part assemblies. Fabrica is the first to demonstrate robust and generalizable performance across a wide range of real-world assembly tasks. We discuss limitations and future work in Sec 7.

7 Limitations and Future Work

While Fabrica shows promising results for autonomous multipart assembly, there remain several limitations and opportunities for future extension.

Assumptions: The assumptions we make in this problem formulation are the following:

(A1) *Insertion-only assembly*: We assume that the mating between two parts only involves an insertion motion, without requiring skills like screwing or sliding.

(A2) *No subassembly reorientation*: The final assembled positions of all parts are assumed to be given and fixed during the assembly process. This means that no further movement or re-orientation is allowed once a part is assembled.

(A3) *Monotonic assembly*: Each part is only moved once, without considering regrasps, in-hand manipulation, or handovers.

(A4) *No force and torque constraints for the robots*: We assume all parts are light compared to the robot payload.

(A5) *A finite grasp set for each part*: $\forall g \in \sigma, g \in \mathcal{G}[o], |\mathcal{G}[o]| = N$, where $\mathcal{G}[o]$ can be computed by any grasp generator.

The above assumptions leave areas such as handling heavier parts, managing grasp slippage, and performing other operations such as screwing or sliding unaddressed. Incorporating these capabilities would significantly improve the robustness and applicability of Fabrica in more complex and diverse assembly tasks.

Dexterity: Moreover, the current setup enforces a fixed part pose once assembled. This is in contrast with the more dexterous human assembly behavior, where one would constantly reorient the partial assembly so that the parts are easily reachable. However, allowing reorientation would introduce additional planning overhead and more uncertainty for control due to potential subassembly instability. Addressing it will enable a more dexterous robotic system that can handle large assemblies that are beyond the reach of the current system, e.g., a large tabletop with parts on both sides.

Hardware capability: Compared to the existing multipart assembly dataset with thousands of objects [10], our current benchmark is limited in its size and diversity. This is because we want to ensure that the benchmark tasks are achievable by commonly used parallel grippers, but these grippers have a limited grasp width and thus cannot establish stable antipodal grasps for parts with large and complex geometries. However, to broaden the assembly capability, we can envision either a multi-finger hand or a multi-tool system in which the robots can switch specialized grippers according to the part geometry, and our planning and control system could be adapted to this setting.

Perception: Integrating vision systems for alignment feedback could greatly improve the accuracy and adaptability of the assembly process. By incorporating perception, the system could enable direct bin-picking, allowing it to grasp parts from random, unknown initial poses instead of requiring a specialized pickup fixture. However, bin-picking remains a well-known challenge in industry, particularly in terms of robustness and generalizability across arbitrary part geometries and physical properties. Addressing these challenges requires substantial research and development efforts, but would significantly expand the practical applications of our approach.

Data collection: Collecting real-world assembly data is challenging due to the task's long-horizon, contact-rich nature and the high cost of acquiring or fabricating diverse assembly assets. Fabrica addresses these barriers by enabling fully autonomous data collection in simulation and requiring only minimal human intervention in the real world. In future work, we aim to leverage this capability to generate large-scale, diverse datasets of assembly trajectories. These datasets can facilitate broader research in generalizable policy learning, sim-to-real transfer, and foundation models for robotic manipulation. Moreover, because assembly is one of the most constrained and demanding manipulation tasks, learning from assembly data has the potential to positively transfer to a wide range of general-purpose manipulation skills.

Acknowledgments

This work is funded by Autodesk, and in part by NSF 1846368 & 2313076. Yijiang Huang is supported by the SNSF Ambizione program. The authors would like to thank Bingjie Tang and Lars Ankile for insightful discussions during the RL environment setup, Xiang Zhang and Yotto Koga for valuable advice on the early physical setup, the members of Ted Adelson's lab at MIT for their support with the physical Panda robot, and the MIT SuperCloud and Lincoln Laboratory for HPC resources.

References

[1] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda. Efficient tactile simulation with differentiability for robotic manipulation. In *6th Annual Conference on Robot Learning*, 2022.

[2] X. Zhang, M. Tomizuka, and H. Li. Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion. In *ICRA*, 2024.

[3] B. Tang, I. Akinola, J. Xu, B. Wen, A. Handa, K. Van Wyk, D. Fox, G. S. Sukhatme, F. Ramos, and Y. Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.

[4] D. Halperin, J.-C. Latombe, and R. H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26(3):577–601, 2000.

[5] S. Sundaram, I. Remmler, and N. M. Amato. Disassembly sequencing using a motion planning approach. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1475–1480. IEEE, 2001.

[6] D. T. Le, J. Cortés, and T. Siméon. A path planning approach to (dis) assembly sequencing. In *2009 IEEE International Conference on Automation Science and Engineering*, pages 286–291. IEEE, 2009.

[7] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga. C-space tunnel discovery for puzzle path planning. *ACM Transactions on Graphics (TOG)*, 39(4):104–1, 2020.

[8] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters. Learn2assemble with structured representations and search for robotic architectural construction. In *Conference on Robot Learning*, pages 1401–1411. PMLR, 2022.

[9] S. K. S. Ghasemipour, S. Kataoka, B. David, D. Freeman, S. S. Gu, and I. Mordatch. Blocks assemble! learning to assemble with large-scale structured reinforcement learning. In *International Conference on Machine Learning*, pages 7435–7469. PMLR, 2022.

[10] 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 Transactions on Graphics (TOG)*, 41(6):1–11, 2022.

[11] Y. Tian, K. D. Willis, B. Al Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. 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.

[12] I. Rodrıguez, K. Nottensteiner, D. Leidner, M. Kaßecker, F. Stulp, and A. Albu-Schäffer. Iteratively refined feasibility checks in robotic assembly sequence planning. *IEEE Robotics and Automation Letters*, 4(2):1416–1423, 2019.

[13] X. Zhu, D. K. Jha, D. Romeres, L. Sun, M. Tomizuka, and A. 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.

[14] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham. Can robots assemble an ikea chair? *Science Robotics*, 3(17):eaat6385, 2018.

[15] L. Nägele, A. Hoffmann, A. Schierl, and W. Reif. Legobot: Automated planning for coordinated multi-robot assembly of lego structures. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9088–9095. IEEE, 2020.

[16] Y. Huang, C. R. Garrett, I. Ting, S. Parascho, and C. T. Mueller. Robotic additive construction of bar structures: Unified sequence and motion planning. *Construction Robotics*, 5:115–130, 2021.

[17] Y. Huang, P. Y. V. Leung, C. Garrett, F. Gramazio, M. Kohler, and C. Mueller. The new analog: A protocol for linking design and construction intent with algorithmic planning for robotic assembly of complex structures. In *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication*, pages 1–17, 2021.

[18] Z. Wang, F. Kennel-Maushart, Y. Huang, B. Thomaszewski, and S. Coros. A temporal coherent topology optimization approach for assembly planning of bespoke frame structures. *ACM Transactions on Graphics (TOG)*, 42(4):1–13, 2023.

[19] C.-J. Liang, S.-C. Kang, and M.-H. Lee. Ras: a robotic assembly system for steel structure erection and assembly. *International Journal of Intelligent Robotics and Applications*, 1:459–476, 2017.

[20] K. Dörfler, T. Sandy, M. Giftthaler, F. Gramazio, M. Kohler, and J. Buchli. Mobile robotic brickwork: automation of a discrete robotic fabrication process using an autonomous mobile robot. *Robotic Fabrication in Architecture, Art and Design 2016*, pages 204–217, 2016.

[21] A. A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, and M. Kohler. Robotic assembly of timber joints using reinforcement learning. *Automation in Construction*, 125:103569, 2021.

[22] A. Kramberger, A. Kunic, I. Iturrate, C. Sloth, R. Naboni, and C. Schlette. Robotic assembly of timber structures in a human-robot collaboration setup. *Frontiers in Robotics and AI*, 8: 768038, 2022.

[23] 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)*, pages 3524–3531. IEEE, 2018.

[24] Y. Fan, J. Luo, and M. Tomizuka. A learning framework for high precision industrial assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 811–817. IEEE, 2019.

[25] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, et al. Factory: Fast contact for robotic assembly. *arXiv preprint arXiv:2205.03532*, 2022.

[26] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. *arXiv preprint arXiv:2305.17110*, 2023.

[27] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka. Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning. In *Conference on Robot Learning*, pages 1621–1639. PMLR, 2023.

[28] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. *arXiv preprint arXiv:2401.16013*, 2024.

[29] M. Noseworthy, B. Tang, B. Wen, A. Handa, N. Roy, D. Fox, F. Ramos, Y. Narang, and I. Akinola. Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty. *arXiv preprint arXiv:2408.04587*, 2024.

[30] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. *arXiv*, 2024.

[31] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

[32] H. Huang, D. Wang, A. Tangri, R. Walters, and R. Platt. Leveraging symmetries in pick and place. *The International Journal of Robotics Research*, page 02783649231225775, 2024.

[33] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[34] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. In *Conference on Robot Learning*, pages 835–846. PMLR, 2023.

[35] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. *arXiv preprint arXiv:2310.16050*, 2023.

[36] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.

[37] J. Seo, N. P. Prakash, X. Zhang, C. Wang, J. Choi, M. Tomizuka, and R. Horowitz. Contact-rich se (3)-equivariant robot manipulation task learning via geometric impedance control. *IEEE Robotics and Automation Letters*, 2023.

[38] W. Lian, T. Kelch, D. Holz, A. Norton, and S. Schaal. Benchmarking off-the-shelf solutions to robotic assembly tasks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1046–1053. IEEE, 2021.

[39] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on robot learning*, pages 767–782. PMLR, 2018.

[40] Y. Lee, E. S. Hu, and J. J. Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. In *2021 ieee international conference on robotics and automation (icra)*, pages 6343–6349. IEEE, 2021.

[41] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[42] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *arXiv preprint arXiv:2401.08553*, 2024.

[43] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. In *WAFR Proceedings of the workshop on Algorithmic foundations of robotics*, pages 109–125. AK Peters, Ltd. Natick, MA, USA, 1994.

[44] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746, 2004.

[45] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[47] K. Doshi, M. Bagatella, and S. Coros. Problem space transformations for generalisation in behavioural cloning. *arXiv preprint arXiv:2411.04056*, 2024.

[48] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[49] 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, 2021.

[50] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2021.

[51] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252, 2002.

[52] J. Jylänki. A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing, 2010. URL https://raw.githubusercontent.com/juj/RectangleBinPack/master/RectangleBinPack.pdf.

Appendix A Qualitative Results

Fig. 1 and Fig. 2 show qualitative results of our system assembling a variety of multi-part objects using different robot arms, both in simulation and in the real world. Once the hardware configuration is provided, our planner works seamlessly with a wide range of robot arms and end-effectors without requiring additional tuning.

The assembly process begins with picking up parts from the fixture, then transferring them to the assembly area for either holding or insertion. The system determines when to switch roles between the two arms, for example by handing over a part or changing the holding grasp, in order to maintain stability and accuracy. After each insertion, the robot returns to retrieve the next part, and this process continues until the assembly is complete. Finally, both arms return to their initial positions.

All plans are generated automatically, including grasp selection, arm coordination, motion generation, and fixture design. The only manual input required is the initial setup of the workcell and placement of the hardware.

For real-world execution, we demonstrate robust sim-to-real transfer across long-horizon assembly tasks, with consistent step-by-step correspondence between simulation and physical execution. The system maintains geometric and temporal alignment across all stages of the assembly, including grasping, insertion, and part switching. This highlights the reliability of our simulation-informed planning and policy execution.

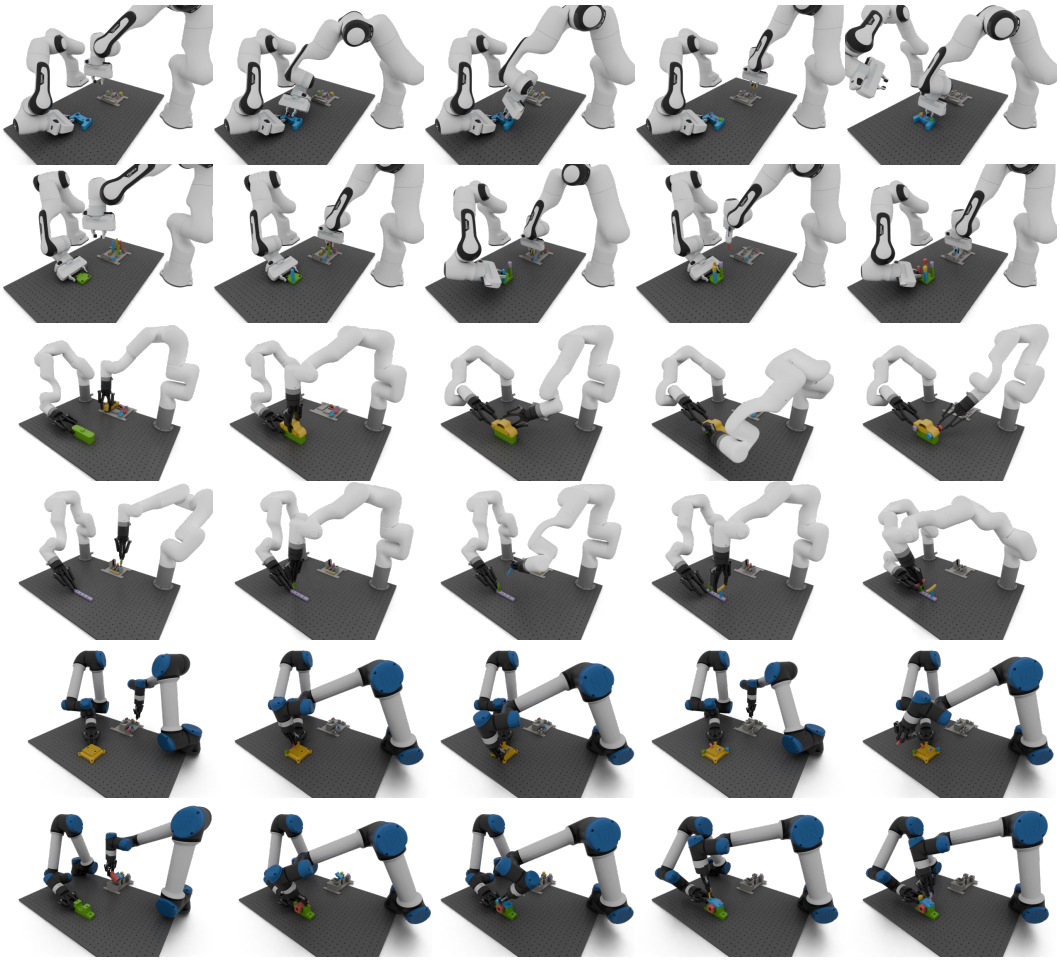


Figure 1: Step-by-step rendered assembly executions on different assemblies with different robots.

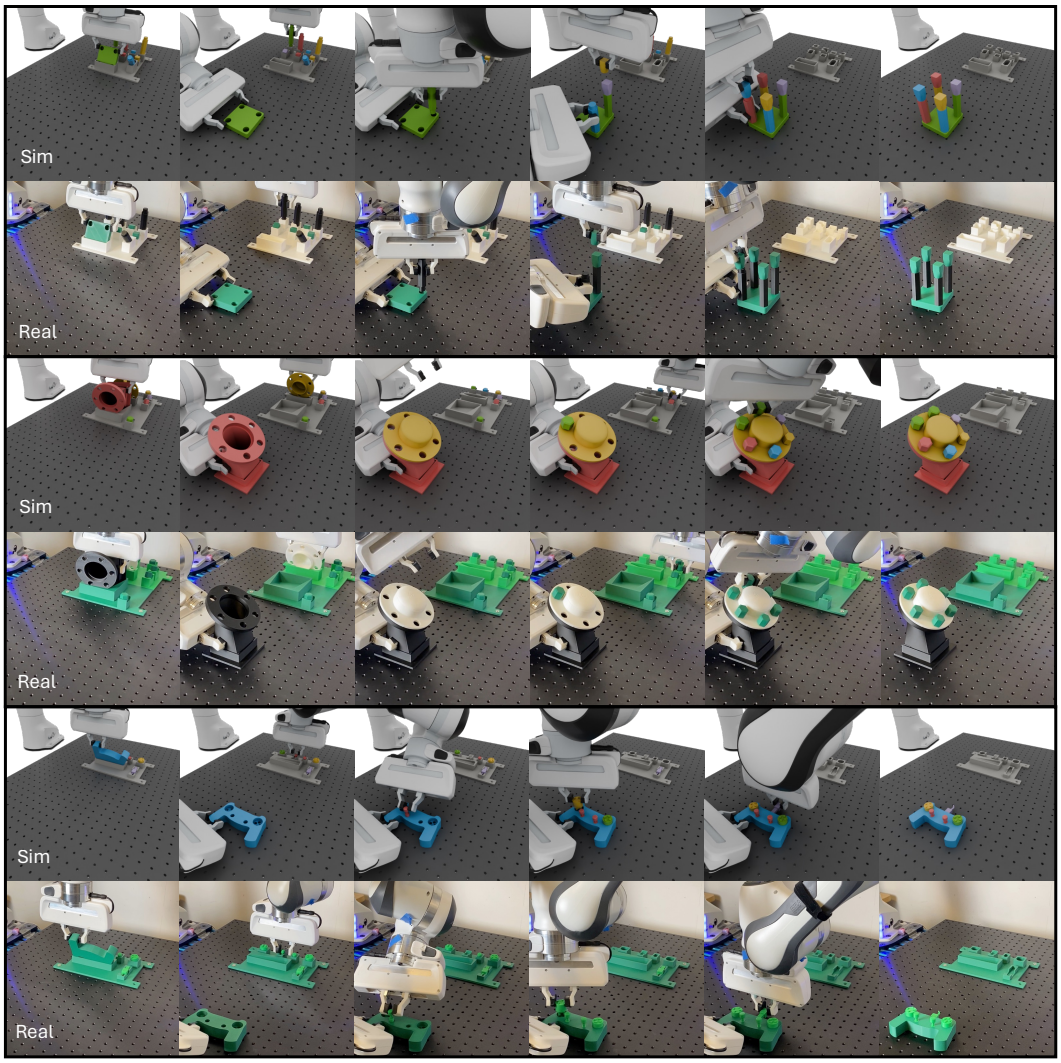


Figure 2: Step-by-step real-world assembly executions on different assemblies with Panda robots, with side-by-side correspondences between simulation and real.

Appendix B Problem Formulation

As in Sec. 3, we formulate planning as optimizing assembly-hold sequences ϕ, grasps σ, and robot motions π:

$$\min_{\phi,\sigma,\pi} E\big(\Phi_{i=1}^{n}\vec{f}(\phi_{1:i},\sigma_{i-1:i},\pi_i)\big) \quad \text{s.t.} \quad C_{\text{prec}}(\phi)\leq 0,\ C_{\text{kin}}(\phi,\sigma,\pi)=0,\ C_{\text{col}}(\phi,\sigma,\pi)\leq 0 \qquad (2)$$

A more detailed breakdown of constraints $C_{\text{prec}}(\phi)$, $C_{\text{kin}}(\phi)$, $C_{\text{col}}(\phi)$ is shown below.

$$\begin{aligned}
&\min_{\phi,\sigma,\pi} \ \vec{F}(\phi,\sigma) \\
&\text{s.t.} \quad \forall i \in [1,n], \\
&\qquad C_{\text{prec}}(\phi_{1:i}) \leq 0 && \text{(3a)} \\
&\qquad C_{\text{kin}}(o_{a,i}, g_{a,i}, \tau^{g}_{a,i}(0), p^{0}_{o_i}) = 0 && \text{(3b)} \\
&\qquad C_{\text{kin}}(o_{a,i}, g_{a,i}, \tau^{f}_{a,i}(1), p^{G}_{o_i}) = 0 && \text{(3c)} \\
&\qquad C_{\text{kin}}(o_{h,i}, g_{h,i}, \tau^{f}_{h,i}(1), p^{G}_{o_i}) = 0 && \text{(3d)} \\
&\qquad C_{\text{col}}(\phi_{1:i}, \pi[a,i], \pi[h,i]) \leq 0 && \text{(3e)}
\end{aligned}$$

where each motion $\tau : [0,1] \to \mathcal{Q}$ is a time-parametrized joint trajectory in the robot's configuration space $\mathcal{Q}$. Eq. (3a) are the precedence constraints that ensure the partially assembled parts are connected and do not collapse under gravity. Eqs. (3b) to (3d) ensure the robot configuration, grasp, and the grasped object pose are kinematically consistent when picking, assembling, and holding. Eq. (3e) ensures that both robots' trajectories do not collide with previously assembled parts and other static obstacles.

The key planning stages in Sec. 3 are further summarized here.

- Sec. 3.1: Starting from a multi-part mesh model, we construct a precedence graph representing a minimum constraint set for any feasible sequence, considering only collision among parts.
- Sec. 3.2: To reduce online collision checking overhead during search, we pre-compute a discretized, collision-free grasp set for assembling and holding for all part pairs. Each feasible grasp is associated with a corresponding robot trajectory.
- Sec. 3.3: We leverage the precedence graph and precomputed grasp pairs to expand a state tree that contains all feasible part-grasp sequences and then search for an optimal part-grasp sequence that minimizes a grasp stability cost.
- Sec. 3.4: After the grasps are determined, we develop an automatic design algorithm to generate a pickup fixture, so that the planned grasp can be achieved easily without the need of a re-grasp.
- Sec. 3.5: With the assembly sequence fixed and all robot configurations determined for kinematic switches, i.e., pick-up, assembly, and hold, we plan for all transit and transfer motions.

Appendix C Algorithmic Details

We now present detailed mathematical formulations of each algorithm introduced in Sec. 3.

C.1 Part Precedence Planning

Alg. 1 and the following paragraphs provide the details of the part precedence planning algorithm.

Precedence Tier Generation Initially, an empty list of tiers T_{prec} is initialized. We use O_r to represent all the remaining assembled parts, which starts from all parts O. Although O_r is not empty, the algorithm constructs a new tier t_{prec} by evaluating each part $o_i \in O_r$ to determine the feasibility of disassembly. Using motion planning, a disassembly path q_i is computed for o_i. Specifically, we apply Assemble-Them-All [10], a physics-based method for efficient disassembly motion planning given the highly constrained search space, and determine success based on a given timeout. If o_i can be disassembled, the pair (o_i, q_i) is added to t_{prec}. Once all feasible parts are processed, the constructed tier t_{prec} is added to T_{prec}, and all $o_i \in t_{\text{prec}}$ is removed from O_r. This process repeats until all parts are assigned to tiers.

Precedence Graph Generation The graph construction phase generates a directed graph G_{prec} that encodes precedence constraints among parts. An empty set O_e is initialized to track parts in earlier tiers (i.e., parts that are supposed to be disassembled earlier). For each tier $t_{\text{prec}} \in T_{\text{prec}}$, the algorithm processes every part $o_i \in t_{\text{prec}}$ by checking its disassembly path τ_{o_i} for collisions with parts in earlier tiers O_e. A collision check function identifies the set of colliding parts $O_c \subseteq O_e$. For each $o_c \in O_c$, an edge (o_i, o_c) is added to G_{prec}, indicating that o_i's disassembly depends on o_c's prior removal. After processing all parts in t_{prec}, the disassembled parts O_t are added to O_e. The algorithm proceeds until all parts in all precedence tiers are added to the precedence graph.

Algorithm 1 Part Precedence Planning

Input: All parts O with goal poses p_O^G
Output: Directed graph G_{prec} representing precedence constraints
Initialize an empty list of tiers: $T_{\text{prec}} \leftarrow$ []
Initialize an empty directed graph: $G_{\text{prec}} \leftarrow \text{DiGraph}()$
Initialize all remaining parts $O_r \leftarrow O$
while $O_r \neq \emptyset$ **do** ▷ Tier generation
    Initialize an empty tier: $t_{\text{prec}} \leftarrow \{\}$
    for each part $o_i \in O_r$ **do**
        $\tau_{o_i} \leftarrow \text{DisassemblyPath}(o_i, O_r)$
        if feasible τ_{o_i} is found **then**
            $t_{\text{prec}} \leftarrow t_{\text{prec}} \cup \{(o_i, \tau_{o_i})\}$
    $O_r \leftarrow O_r \setminus \{o_i \mid (o_i, \tau_{o_i}) \in t_{\text{prec}}\}$
    $T_{\text{prec}}.\text{Append}(t_{\text{prec}})$
Initialize an empty set of parts in earlier tiers: $O_e \leftarrow \emptyset$
for each tier $t_{\text{prec}} \in T_{\text{prec}}$ **do** ▷ Graph construction
    Let $O_t \leftarrow \{o_i \mid (o_i, \tau_{o_i}) \in t_{\text{prec}}\}$ ▷ Parts in tier t_{prec}
    for each $(o_i, \tau_{o_i}) \in t_{\text{prec}}$ **do**
        $G_{\text{prec}}.\text{AddNode}(o_i, \text{path} = \tau_{o_i})$
        $O_c \leftarrow \text{CheckPathCollision}(o_i, \tau_{o_i}, O_e)$ ▷ O_c: colliding parts in O_e
        for each $o_c \in O_c$ **do**
            $G_{\text{prec}}.\text{AddEdge}(o_i, o_c)$
    $O_e \leftarrow O_e \cup O_t$ ▷ Update parts in earlier tiers
return G_{prec}

C.2 Dual-Arm Grasp Filtering

This section provides a detailed description of the sub-steps involved in the grasp filtering algorithm.

Single-Pose Grasp Feasibility Check Alg. 2 evaluates whether a specific grasp configuration g is feasible for a target part o in its current pose p_o. The algorithm first determines the set of preceding parts O_{prec} from the precedence graph G_{prec}. For each gripper aperture (a_{grasp} and a_{release}), collision checks are performed involving the robot body, the gripper, the target part o, the preceding parts O_{prec}, and the environment obstacles E. If any collision occurs between them, the grasp g is not feasible. Additional collision checks are performed between the gripper and other non-preceding parts $O_{\text{all}} \setminus O_{\text{prec}}$, and collision information is added to the grasp. Such collisions are not hard constraints because the non-preceding parts may get assembled later than the target part depending on the specific assembly sequence. In this case, the collision does not matter, but the information has to be recorded to find collision-free assembly sequences in the later stage.

Assembling Grasp Feasibility Check Alg. 3 determines the feasibility of using a grasp g to disassemble a target part o along its disassembly path τ_o, derived from G_{prec}. For each pose $p_{o,t}$ along τ_o, the grasp is transformed accordingly and its feasibility is validated using Alg. 2. The aggregated collision and IK information across all poses is stored in g. The grasp is feasible if all poses along τ_o are validated.

Algorithm 2 Single-Pose Grasp Feasibility Check

Input: Grasp g, robot R, target part o with pose p_o, all parts $o \in O$ with their corresponding goal poses p_o^G, precedence graph G_{prec}
Output: Feasibility (**True/False**), updated grasp g with collision and IK information
$O_{\text{prec}} \leftarrow G_{\text{prec}}.\text{PrecedingParts}(o)$
$E \leftarrow$ environment obstacles
$q_g^R \leftarrow \text{IK}(R, g, o, p_o)$ ▷ IK for robot to grasp part o under pose p_o with grasp g
if feasible q_g^R is found **then**
    Set robot R configuration to q_g^R
    for each gripper aperture $a \in \{a_{\text{grasp}}, a_{\text{release}}\}$ **do**
        Set gripper with aperture a
        if CheckCollision$(R, o, O_{\text{prec}}, E)$ **then**
            return False, g
        CheckCollision$(R, O \setminus O_{\text{prec}})$
    Record collision and IK information to g
    return True, g
else
    return False, g

Algorithm 3 Assembling Grasp Feasibility Check

Input: Grasp g, robot R, target part o, all parts O with goal poses p_O^G, precedence graph G_{prec}
Output: Feasibility (**True/False**), updated grasp g with collision and IK information
$\tau_o \leftarrow G_{\text{prec}}.\text{GetPath}(o)$ ▷ Disassembly path of part o
for each part pose $p_{o,t} \in \tau_o$ **do** ▷ Disassembling o while grasping o
    Transform grasp g according to p_o^t to obtain g_t
    feas, $g_t \leftarrow$ CheckGraspFeas(g_t, o, $p_{o,t}$, ...) ▷ Alg. 2
    if not feas **then**
        return False, g
    Gather collision and IK information from g_t to g
return True, g

Holding Grasp Feasibility Check Alg. 4 evaluates whether a grasp g can securely hold a part o while allowing other parts o_i to be disassembled. The feasibility of g is first validated using Alg. 2.

Algorithm 4 Holding Grasp Feasibility Check

Input: Grasp g, robot R, target part o, all parts O with goal poses p_O^G, precedence graph G_{prec}
Output: Feasibility (**True/False**), updated grasp g with collision and IK information
feas, $g \leftarrow$ CheckGraspFeas(g, o, p_o^G, ...) ▷ Alg. 2
if not feas **then**
    return False, g
for each part $o_i \in O_{\text{all}} \setminus \{o\}$ **do**
    $q_{o_i} \leftarrow G_{\text{prec}}.\text{GetPath}(o_i)$ ▷ Disassembly path of o_i
    for each part pose $p_{o_i,t} \in q_{o_i}$ **do** ▷ Disassembling o_i while grasping o
        CheckCollision(R, o_i) and gather collision information to g
return True, g

Grasp Pair Filtering Putting the assembling and holding feasibility checks together, Alg. 5 generates and filters the dual-arm grasp pairs. For each part o_i, a set of candidate grasp poses $\{g_k\}_{k=1}^{N_g}$ is generated. Each grasp is evaluated for assembling and holding feasibility using Algs. 3 and 4, respectively, and feasible grasps are stored in $\mathcal{G}^a[o_i]$ and $\mathcal{G}^h[o_i]$. Finally, iterating through all assembly-hold part pairs, the set of feasible assembly-hold grasp pairs $\mathcal{G}^{a\times h}[o_a, o_h]$ only contains those that do not lead to collisions between the two robots.

Algorithm 5 Dual-Arm Grasp Pair Filtering

Input: All parts O with goal poses p_O^G, robots R_a, R_h, precedence graph G_{prec}
Output: Assembling grasps $\mathcal{G}^a$, holding grasps $\mathcal{G}^h$, assembling-holding grasp pairs $\mathcal{G}^{a\times h}$
Initialize empty dictionaries $\mathcal{G}^a, \mathcal{G}^h, \mathcal{G}^{a\times h} \leftarrow \{:\}, \{:\}, \{:\}$
for each part o_i in O **do** ▷ Feasible grasp generation
    $\mathcal{G}^a[o_i], \mathcal{G}^h[o_i] \leftarrow \{\}, \{\}$
    Generate N_g grasp poses $\{g_k\}_{k=1}^{N_g}$ on part o_i
    for each grasp pose g_k **do**
        $\text{feas}_a, g_a \leftarrow \text{CheckAssemGraspFeas}(g_k, R_a, o_i, ...)$
        if feas_a **then**
            $\mathcal{G}^a[o_i] \leftarrow \mathcal{G}^a[o_i] \cup \{g_a\}$
        $\text{feas}_h, g_h \leftarrow \text{CheckHoldGraspFeas}(g_k, R_h, o_i, ...)$
        if feas_h **then**
            $\mathcal{G}^h[o_i] \leftarrow \mathcal{G}^h[o_i] \cup \{g_h\}$
for each part $o_a \in O$ **do** ▷ Feasible grasp pair filtering
    for each part $o_h \in O$ **do**
        if $o_a \in G_{\text{prec}}.\text{PrecedingParts}(o_h)$ **then**
            continue
        $\mathcal{G}^{a\times h}[(o_a, o_h)] \leftarrow \{\}$
        for each grasp $g_a \in \mathcal{G}^a[o_a]$ **do**
            for each grasp $g_h \in \mathcal{G}^h[o_h]$ **do**
                Set robot R_a configuration to $q_{g_a}^{R_a}$
                Set robot R_h configuration to $q_{g_h}^{R_h}$
                if not $\text{CheckCollision}(R_a, R_h)$ **then**
                    $\mathcal{G}^{a\times h}[(o_a, o_h)] \leftarrow \mathcal{G}^{a\times h}[(o_a, o_h)] \cup \{(g_a, g_h)\}$
return $\mathcal{G}^a, \mathcal{G}^h, \mathcal{G}^{a\times h}$

C.3 Dual-arm Sequence-Grasp Optimization

Alg. 6 provides the pseudocode for the sequence-grasp optimization algorithm, followed by the formulas used to evaluate the transition edge cost.

Objective evaluation

- Maximizing the number of supportive parts held (f_1): Part A is supportive to part B if A is in the preceding parts of B in G_{prec}.
- Minimizing the number of holding grasp transitions (f_2): Holding grasp transitions can be counted simply by comparing whether the holding grasps in consecutive steps are the same.
- Minimizing approximated torque for assembling grasps (f_3):

$$\frac{\|\boldsymbol{\tau}_{\text{part}} + \boldsymbol{\tau}_{\text{grasp}}\|}{N_{\text{grasp}}} \tag{4}$$

  Where:

$$\boldsymbol{\tau}_{\text{part}} = \sum_{i=1}^{N_{\text{part}}} (\mathbf{r}_i - \mathbf{c}_{\text{part}}) \times \frac{\mathbf{d}_{\text{contact}}}{N_{\text{part}}}$$

$$\boldsymbol{\tau}_{\text{grasp}} = \sum_{j=1}^{N_{\text{grasp}}} (\mathbf{r}_j - \mathbf{c}_{\text{part}}) \times \frac{-\mathbf{d}_{\text{contact}}}{N_{\text{grasp}}}$$

  Where:

  – $\mathbf{r}_i, \mathbf{r}_j$ are the position vectors of contact points on the part and grasp, respectively.
  – $\mathbf{c}_{\text{part}}$ is the center of mass of the part.

Algorithm 6 Dual-Arm Assembly Sequence Planning

Input: All parts O, precedence graph G_{prec}, assembling grasps $\mathcal{G}^a$, assembling-holding grasp pairs $\mathcal{G}^{a\times h}$
Output: Optimal assembly part-grasp sequence S^*
Initialize an empty tree $T_G \leftarrow$ DiGraph()
S $\leftarrow$ [] ▷ Initialize an empty search stack.
for each $o \in O$ **do**
  if G_{prec}.SucceedingParts$(o) = \emptyset$ **then**
    for each $g \in \mathcal{G}^a[o]$ **do**
      T_G.AddNode$((a, O \setminus \{o\}, o, g))$ ▷ Root nodes
      push(S, $(a, O \setminus \{o\}, o, g)$)
while S **not** empty **do**
  $(t_i, O_i, o_i, g_i) \leftarrow$ **pop**(S)
  $t_{i+1} \leftarrow h$ **if** $t_i = a$ **else** $t_{i+1} \leftarrow a$
  for each $o_{i+1} \in O_i$ **do**
    $O_{i+1} \leftarrow O_i \setminus \{o_i\}$ **if** $t_{i+1} = a$ **else** $O_{i+1} \leftarrow O_i$
    if $(O_i \cup \{o_i\}) \cap G_{\text{prec}}$.PrecedingParts$(o_{i+1}) \neq \emptyset$ **then**
      continue ▷ Precedence check
    for each $g_{i+1} \in \mathcal{G}^{t_{i+1}}[o_{i+1}]$ **do**
      $O_c \leftarrow \mathcal{G}^{t_{i+1}}[o_{i+1}][g_{i+1}]$.CollidingSet ▷ Precomputed collision set
      if $O_c \cap O_{i+1} \neq \emptyset$ **then**
        continue ▷ State collison check
      if $(g_{i+1}, g_i) \in \mathcal{G}^{t_{i+1}\times t_i}[(o_{i+1}, o_i)]$ **then**
        T_G.AddEdge$(((t_i, O_i, o_i, g_i), (t_{i+1}, O_{i+1}, o_{i+1}, g_{i+1})))$ ▷ Grasp pair validity check
        push(S, $(t_{i+1}, O_{i+1}, o_{i+1}, g_{i+1})$)
for each $e_i \in T_G$.Edges() **do** ▷ Objective evaluation
  $((t_i, O_i, o_i, g_i), (t_j, O_j, o_j, g_j)) \leftarrow e_i$
  if $t_i = a, t_j = h$ **then** ▷ Assembling-holding step
    $e_i.\vec{f} \leftarrow \vec{f}(O_i, o_i, g_i, o_j, g_j)$
  else ▷ Holding transition step
    $e_i.\vec{f} \leftarrow \vec{0}$
$S^* \leftarrow [o^*_{a,1}, g^*_{a,1}, o^*_{h,2}, g^*_{h,2}, ..., o^*_{a,n}, g^*_{a,n}] \leftarrow \text{DP}(T_G, \Phi)$ ▷ Dynamic programming
return S^*

  - $\mathbf{d}_{\text{contact}}$ is the contact direction vector.
  - N_{grasp} and N_{part} are the number of contact points for the grasp and part, respectively.

- Maximizing part contact area for holding grasps, weighted by the orientation difference from the paired assembling grasps (f_4):

$$\Delta\theta_{\text{rotation}} \times N_{\text{hold}} \tag{5}$$

Where:

$$\Delta\theta_{\text{rotation}} = \left\|\mathbf{R}_{\text{hold}}^{-1} \cdot \mathbf{R}_{\text{assemble}}\right\|$$

Where:

  - $\Delta\theta_{\text{rotation}}$ represents the angular difference between the holding and assembling grasps.
  - $\mathbf{R}_{\text{hold}}$ and $\mathbf{R}_{\text{assemble}}$ are the rotation matrices derived from the quaternions of the holding and assembling grasps, respectively.
  - N_{hold} is the number of contact points in the holding grasp.

C.4 Grasp-Oriented Pickup Fixture Generation

The fixture generation process begins with the computation of an appropriate pickup pose for each part. Our goal is to enforce a top-down pickup grasp without requiring re-grasping during the transition from pickup to assembly. Therefore, we can derive the pickup orientation of each part given

the final assembled pose of them and the optimal grasp planned at the assembled pose, since we maintain the same relative transformation between the part and the gripper from pickup to assembly.

Next, we determine the pickup position for all parts. Since all pickup motions follow a top-down path, parts are arranged to prevent any overlap along the Z-axis, the vertical direction in the world coordinate frame. This constraint ensures unobstructed pickup paths and simplifies the design of the supporting fixture. Along the Z-axis, parts are positioned at the lowest possible height while ensuring that there are no collisions with the ground based on their orientation. In addition, a minimum base height is maintained for the fixture board to provide structural stability and support.

Determining the XY positions of the parts is more challenging, as the layout directly impacts the fixture area, which should ideally be minimized to reduce material costs for fixture fabrication and maximize the available workspace within the workcell. Additionally, incorrect part layout on the XY plane can lead to potential collisions between the gripper and the remaining parts during pickup. Since the orientation of each object is locked, we can represent each part using a rectangular bounding box of its 2D-projected contour. Then, the problem becomes a 2D bin-packing problem, a classic problem with both heuristic and exact algorithms exist [51]. We use a simple algorithm that iterates through the following phases:

1. We use the Maximal Rectangles algorithm [52] to pack the bounding boxes into an initial bounding area;
2. We check the collisions between the gripper and the precedent parts of the grasped part;
3. If any collisions are detected, the colliding parts are buffered with additional spacing and the bin-packing process is performed again;
4. We increase the bin area once it is not enough to find a packing solution given the increased rectangle sizes.

Once an optimal packing configuration is determined, the fixture is generated by creating mold cavities that accommodate the part shapes. A minimal mold depth is calculated to ensure gravitational stability of part placement, where the Z-axis projection of each part's center of mass lies within the convex hull of the contact points between the fixture and the part. The fixture cavity is generated by projecting the part's 3D geometry onto a 2D plane perpendicular to the pickup direction and extruding it to the calculated depth. Additional cavity is generated by creating free space for the grasp motion for every part based on the gripper geometry and the grasping motion. Finally, the generated fixture is enhanced with countersunk pads to assist in accurate positioning. By automating the entire fixture generation process, our approach provides a flexible and scalable solution for diverse part geometries and assembly sequences.

Appendix D Experimental Setup

D.1 Hardware Setup

We conduct real-world experiments using a dual-arm setup composed of two Franka Emika Panda robots, each equipped with parallel-jaw grippers. The arms are mounted on one side of a shared table, facing the user, and their relative positions are calibrated via a common reference frame. The workspace is divided into a pickup area and an assembly area, both fixed and pre-defined based on the available workspace area. The system uses internal encoders for joint sensing, without external force-torque sensors or visual feedback.

D.2 Simulation Environment

We use RedMax, the same simulator used in Tian et al. [11], for simulation-based planning, and Isaac Gym for reinforcement learning. Grasp feasibility is determined by sampling 100 candidate grasps per part using an antipodal grasp planner, followed by inverse kinematics validation and collision

checking in RedMax. Precedence tier planning uses a physics-based disassembly planner [10] with orthogonal force directions for motion planning.

D.3 Training Configuration

Assembly policies are trained using PPO from the RL Games framework with key hyperparameters for RL presented in Table 1, which we use for all reported RL experiments. Our generalist policies are trained for a maximum of 5×10^7 steps (or equivalently 1500 iterations) with a parallel rollout setup using 1024 environments, which takes less than 2 hours on a single NVIDIA RTX A6000 GPU.

Table 1: Key RL Hyperparameters.

Parameter	**Value**
Algorithm Name	PPO
MLP Units	[256, 128, 64]
MLP Activation	elu
Learning Rate	1e-4
Gamma	0.99
Tau	0.95
Entropy Coefficient	0.003
Gradient Norm	1.0
Horizon Length	32
Minibatch Size	512
Mini Epochs	8
Critic Coefficient	2
KL Threshold	0.016

D.4 Real-World Deployment

Robot control alternates between executing planned transit motions in joint space and reactive policy execution for insertion steps. For policy execution, a task-space impedance controller is used with gains K_p = [800 N/m, 800 N/m, 400 N/m] and $K_d = 2\sqrt{K_p}$, transformed into the path-centric frame to ensure consistent compliance. Control runs at 30 Hz. During deployment, we use the leaky Policy-Level Action Integrator (leaky PLAI) scheme for improved stability with an action scale of 0.001 and an error threshold of 0.02. We use the same set of control parameters across all benchmark assemblies. Interventions are requested after three consecutive failures to insert, detected via joint deviation thresholds and motion stagnation.

Appendix E Ablation Studies

E.1 Impact of Coordinate and Action Design

We conducted ablation studies to evaluate the impact of coordinate frame selection (world vs. path-centric) and action representation (nominal vs. residual) on the assembly specialist (AS) policy's performance in simulation. Table 2 presents the average % of successful steps without intervention across 1024 randomized simulation evaluations for each assembly task under four configurations.

Overall, the combination of path-centric coordinates and residual actions proves particularly effective for assemblies involving diverse insertion directions, such as the Plumbers Block and Gamepad. Individually, each technique provides a structured prior that simplifies learning: path-centric coordinates standardize insertion directions by reorienting each assembly motion into a canonical frame, while residual actions leverages the planned trajectory and allow for fine-grained corrective adjustments on top of it. When applied together, they complement each other, enabling both directional consistency and precise control, leading to significantly higher success rates across most assemblies in our benchmark.

Table 2: Ablation studies on the impact of coordinate and action design choices on the assembly specialist (AS) policy's performance across 1024 randomized simulation evaluations.

Coordinate	Action	% of Successful Steps without Intervention (Simulation)						
		Beam	Plumber Block	Car	Gamepad	Cooling Manifold	Duct	Stool
World	Nominal	**99.71%**	49.32%	**73.24%**	76.95%	**94.92%**	**96.97%**	70.41%
World	Residual	**99.71%**	95.02%	71.09%	73.44%	**95.02%**	**97.75%**	74.80%
Path-Centric	Nominal	**99.71%**	93.95%	60.35%	85.35%	92.58%	**97.17%**	70.41%
Path-Centric	Residual	**99.12%**	**97.46%**	70.12%	**88.87%**	**95.02%**	**96.58%**	**76.66%**

Table 3: Scaling effect of number of policy trials across 3 complete end-to-end multi-step real-world evaluations.

Method	# Trials	% of Successful Steps without Intervention (Real World)						
		Beam	Plumber Block	Car	Gamepad	Cooling Manifold	Duct	Stool
Open-Loop Tracking	1	42%	25%	20%	20%	17%	14%	21%
Assembly Specialist Policy (AS)	1	58%	58%	60%	40%	67%	43%	75%
	2	67%	58%	80%	60%	67%	79%	75%
	3	75%	67%	87%	73%	83%	79%	88%

E.2 Effect of Number of Trials

In real-world experiments, we evaluate the assembly specialist policy with varying numbers of allowed trials (1, 2, or 3) per step, due to the stochastic nature of the policy. Table 3 summarizes the average % of successful steps without intervention across three end-to-end multi-step assembly runs. Results indicate that permitting additional trials substantially boosts success rates. The policy consistently improves as trial count increases, suggesting that the policy benefits from repeated attempts to refine alignment and correct minor positional errors.

The results further demonstrate open-loop baseline's inability to recover from failures and adapt to variations in the real world environment, even though the planned path is accurate and the real robot is well calibrated. In contrast, the RL policy consistently demonstrates improved progress, with notable gains observed as the number of trials increases. The results show a clear trend: with 1 trial per step, the policy can achieve moderate progress but still faces challenges in most of the assemblies. Introducing retries significantly enhance progress, enabling the system to correct minor errors and overcome small disturbances.

Appendix F Integrating Vision Feedback: VLM for Insertion Alignment

To further address insertion misalignments observed during the initial insertion attempt, we integrate a vision-language model (VLM) to provide corrective alignment feedback in the form of discrete actions. The model, a state-of-the-art version of Gemini (`gemini-2.5-pro-preview-03-25`), is leveraged to assess spatial alignment between the grasped part and the insertion hole based on visual input.

While training visuomotor policies directly from visual input is a common approach for alignment tasks, it requires extensive data collection, task-specific training, and continuous fine-tuning to generalize across diverse parts and insertion scenarios. In contrast, leveraging a VLM for alignment feedback offers significant advantages. VLMs are pre-trained on diverse visual contexts, enabling them to generalize across varied geometries and occlusions without extensive task-specific data collection. Additionally, they provide interpretable, discrete corrective actions (e.g., "move right") accompanied by concise explanations, enhancing both robustness and transparency in alignment tasks, especially under low-cost, fixed-focus camera setups.

F.1 Physical Setup

The vision integration requires only RGB input without high imaging quality, allowing for the use of low-cost cameras. We utilize an Arducam B0205 USB camera ($34.99), shown in Fig. 3, mounted on the robot wrist at an angle optimized to capture the insertion area. The camera is equipped with an IR-CUT filter and infrared LEDs for low-light conditions, but lacks focus adjustment, resulting in reduced image clarity when the insertion part is out of focus.

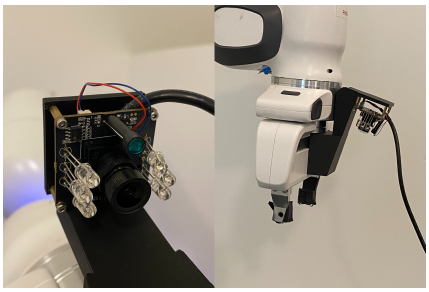


Figure 3: Physical setup for integerating vision feedback. Left: Camera details. Right: The mounted configuration on the robot wrist.

F.2 Methodology

If the insertion policy fails on the first attempt, the entire video of the failed attempt is recorded and segmented into ten key frames sampled uniformly across the video. These frames are passed to the VLM, which is prompted to recommend the best corrective action (`up`, `down`, `left`, or `right`) to align the part with the hole. The VLM is additionally prompted to provide a concise, step-by-step reasoning for the recommended action to mitigate potential hallucinations. The exact prompt we use is detailed below.

```
'''
You are assisting a robot in aligning a grasped part for insertion
    using visual feedback from a camera mounted on the robot's wrist.


Task:
- The part is grasped by the robot and can move in four directions: ["
    up", "down", "left", "right"], each by 2 mm in the camera frame.
- The goal is to move the part to align it precisely with the hole for
     insertion.


Instructions:
- Carefully observe the video frames. Focus only on the position of
    the part relative to the hole.
- Determine the single best action to move the part to align with the
    hole.
- Focus only on spatial cues: Is the part too far left, right, above,
    or below the hole?


Response format:
{
"action": "right",
"reason": "The part is too far left relative to the hole and needs to
    move right to align."
}


Only output the single best action based on spatial cues. If the part
    is already aligned, output "hold".
What is the best action to move the part to align with the hole?
'''
```

After receiving the VLM feedback, the robot executes the recommended action by adjusting the gripper in the task frame by 2 mm. The insertion policy is then restarted from this new position. If the insertion still fails, the entire process is repeated with the newly recorded video sequence, allowing for multiple VLM interventions as necessary.

F.3 Results and Analysis

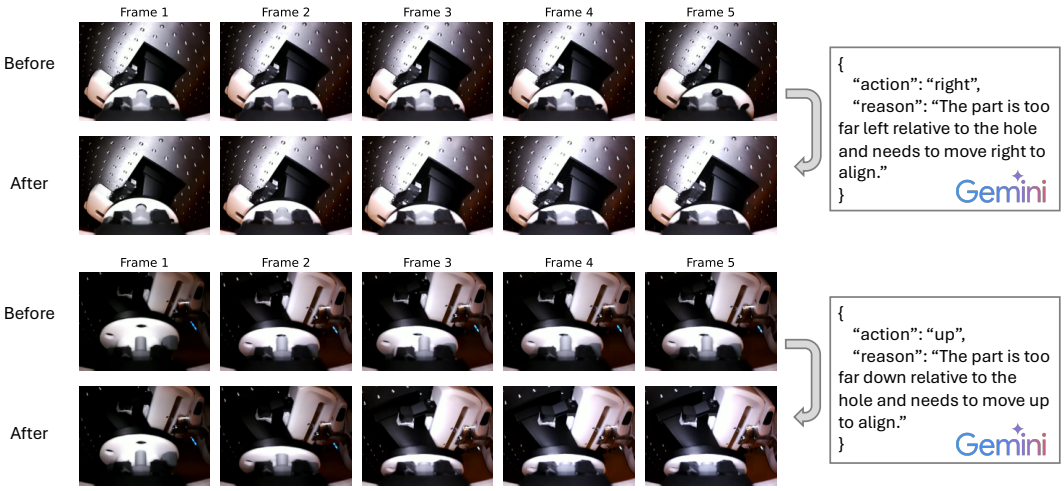


Figure 4: Example outputs from VLM during corrective alignment. The VLM identifies spatial misalignments in the camera frame and recommends discrete corrective actions (e.g., "right" and "up") with concise reasoning.

The VLM integration demonstrated notable improvements in alignment accuracy, particularly in cases where the insertion policy initially failed due to misalignment. Figure 4 illustrates two representative examples where the VLM provided corrective actions that successfully guided the arm to the intended alignment.

In the first example, the VLM identified a leftward misalignment and recommended the action "right", allowing the part to be re-centered relative to the insertion hole. The corrective action was executed in the tool frame, resulting in a more precise alignment before the subsequent insertion attempt. Similarly, in the second example, the VLM detected a downward offset and suggested the action "up", effectively repositioning the part closer to the target insertion point. In both cases, a single VLM intervention was sufficient to resolve the misalignment, highlighting the model's capacity to reason spatially based on minimal visual input.

Despite the low-cost camera setup and lack of focus adjustment capabilities, the VLM effectively discerned alignment cues based on coarse visual features. This is particularly noteworthy given that occlusions and visual clutter are prevalent in multi-part assemblies, where small positional errors can accumulate over successive steps. The VLM's concise, reason-based output structure further mitigates hallucination risks by constraining the response format to a single action-reason pair, reducing ambiguity and enhancing interpretability.

F.4 Limitations and Future Work

While the VLM integration demonstrated significant alignment improvements, occlusions remained a primary failure mode. Occlusions are common in complex assemblies, necessitating either a flexible/active camera setup or multiple cameras strategically positioned to cover the scene comprehensively. Furthermore, hallucination remains a concern, particularly in cluttered scenes where visual cues are ambiguous. Future work will explore improving prompt engineering and camera configurations, potentially leveraging multi-view setups and active camera movements akin to human head and body movements.