

FAST: Action-aware Feasibility in Assembly Sequence Planning using Graph Transformer

Xiaomei Xu[‡], Jaykumar Bhagiya[‡], Marco Schneider, Rainer Müller

Abstract—Automating assembly sequence planning (ASP) for complex mechanical products is difficult due to combinatorial search and physical constraints. Most learning-based ASP methods use an assembly-by-disassembly strategy, selecting parts to remove and relying on physics simulation to test kinematic and structural feasibility—an inefficient, trial-and-error process with limited learning from failures. This work proposes FAST, a feasibility-aware sequence planner that learns to predict the physical feasibility of action-conditioned disassembly operations directly. Using the current geometry and a candidate disassembly action, a Graph Transformer predicts feasibility based on labels from a physics-based checker of assemblability and stability. Trained on a large simulated dataset covering both successful and failed actions, FAST improves planning success by about 6 percentage points under low and 13 percentage points under high computational budgets compared to a learning baseline. Integrated with a collision-free trajectory planner, FAST successfully executes in a real robotic assembly setup, demonstrating end-to-end feasibility from high-level sequence planning to low-level motion control. Github link <https://github.com/Zema-gmbH/FAST>.

I. INTRODUCTION

Modern manufacturing is increasingly dominated by high-mix, low-volume (HMLV) production, where frequent product changes and diverse variants stress traditional automation strategies [1]. In this regime, manually engineering task-specific assembly programs does not scale, and rigid, fixture-heavy automation struggles to reconcile flexibility with throughput. Consequently, there is strong demand for robotic systems that can, given only a product’s CAD model, autonomously derive physically valid assembly sequences as a critical enabler of next-generation autonomous factories [2].

Assembly Sequence Planning (ASP) is, however, fundamentally challenging due to its combinatorial search space and strict physical constraints [3], [4]. Learning-based methods such as reinforcement learning can, in principle, handle combinatorial search but remain sample-inefficient and are mostly demonstrated on small or highly idealised assemblies [5]. From a physical standpoint, the feasibility of a disassembly step depends jointly on geometric accessibility and global structural stability and must be re-evaluated for each candidate action. In standard assembly-by-disassembly

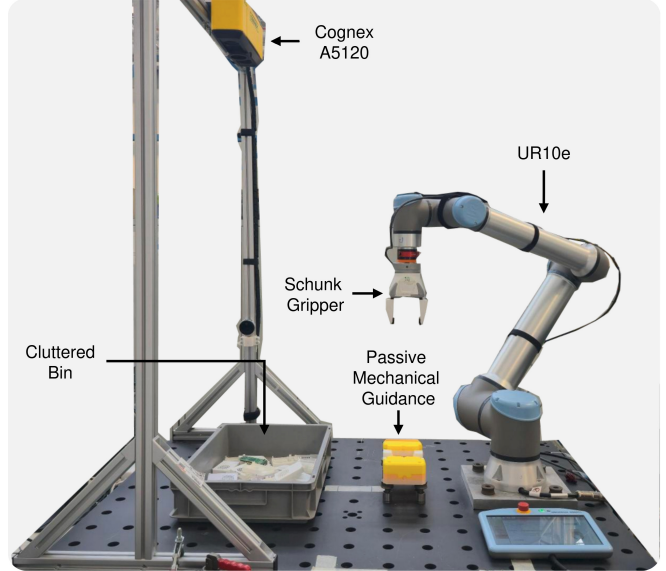


Fig. 1. Overview of the complete robotic assembly system.

formulations [6], [7], each decision requires simulating one or more candidate removals to check for collisions and to test whether the remaining structure collapses or violates static equilibrium. This leads to three practical bottlenecks: (i) heavy reliance on trial-and-error simulation loops, which quickly becomes prohibitive for large assemblies and tight evaluation budgets; (ii) weak learning signals, because simulators usually return only a binary success/failure outcome without explaining whether infeasibility arose from obstruction or loss of support; and (iii) a mismatch between local part-selection policies and the inherently global nature of mechanical constraints, especially in assemblies where removing a seemingly local component destabilises distant regions.

Motivated by these limitations, we reframe ASP as learning an action-conditioned feasibility model: given the current assembly and a candidate disassembly action (removing a part along a specified direction), the model predicts whether executing that action will succeed under full geometric and physical constraints. This makes feasibility itself the learning target, enabling the model to learn from both successful and failed simulation outcomes and to serve as a surrogate for expensive physics-based checks. We instantiate this formulation with FAST, a Graph Transformer-based feasibility model integrated into a disassembly-to-assembly search procedure, in which FAST scores candidate disassembly actions given the assembly state to prioritise evaluation within a limited physics-simulation budget. Fig. 1 gives a high-level overview

This research was funded by the project “IfaR-Saar – Innovationszentrum für angewandte, intelligente Robotik,” supported by the Saarland Ministry of Economic Affairs, Innovation, Digitalisation and Energy (MWIDE)

[‡]These authors share first authorship.

Centre for Mechatronics and Automation Technology gGmbH (ZeMA). Eschberger Weg 46, Saarbrücken, Germany {xiaomei.xu, jaykumar.bhagiya, marco.schneider, rainer.mueller}@zema.de

of the robotic system, covering perception and robotic execution, while the planning core is detailed in Sec . III.

Our main contributions are:

- 1) A feasibility-aware reformulation of ASP as action-conditioned binary prediction, integrated into a disassembly-to-assembly planner.
- 2) A Graph Transformer-based feasibility model on contact graphs that jointly encodes part geometry, physical properties, and action parameters for long-range stability and accessibility reasoning.
- 3) A comprehensive evaluation on a standard physics-based ASP benchmark, using disassembly trees and action labels derived from simulation, with ablations and low-/high-budget analyses.
- 4) Integration of the learned feasibility model into a full robotic assembly pipeline with perception and collision-free motion planning, demonstrating physically valid sequences on a real robot.

II. STATE OF THE ART

A. Assembly Sequence Planning

Early work on robotic assembly and insertion tasks demonstrated that learning can assist in structured domains such as furniture assembly, Lego construction, and peg-in-hole style insertion [8]–[11]. These systems typically operate on fixtures or regular grid-like geometries and either neglect gravity or treat contacts in a highly simplified way, thereby limiting their relevance to industrial assemblies with tight clearances and complex interlocking parts. Beyond learning, classical optimisation methods, most notably genetic algorithms for disassembly sequencing [12], [13], have long been applied to this problem and remain standard baselines, which we include in our evaluation.

More recent assembly-by-disassembly planners such as Assemble Them All and ASAP combine graph neural networks (GNNs) with physics simulation [6], [7]. A GNN predicts a distribution over parts to remove, and a physics engine then checks assemblability (kinematic removability) and stability of the top candidates. While this architecture scales to more complex CAD assemblies, it retains a propose–then–validate loop: if the highest-ranked part fails validation, the planner must backtrack and test further candidates. Under tight simulation budgets, this leads to many wasted evaluations, and the binary success/failure feedback offers little information about why a particular removal was infeasible.

Several works have tried to inject more physics awareness into this loop. Physics-aware action masking for combinatorial ASP uses analytic constraints and simulation heuristics to prune obviously impossible actions before search [14]. This reduces the candidate set but still treats feasibility as a hard constraint checked outside the learned model, and it does not expose fine-grained supervision about stability or contact interactions that could train a richer predictor.

In parallel, Transformer-based sequence models on part graphs have been explored for 3D assembly generation and

planning [15]–[17]. These approaches exploit global self-attention to capture long-range structural dependencies and have shown strong results in generating plausible assembly orders or structures under geometric constraints. However, they are typically evaluated in gravity-free or simplified physical settings and optimise for geometric plausibility rather than explicit physical feasibility; stability and contact mechanics are not enforced through high-fidelity simulation.

A closely related line of work learns removability or success probabilities to speed up ASP [18]. These methods share the goal of reducing simulation calls, but often score parts independently of the full assembly context or direction and do not distinguish between failures caused by collisions and those caused by loss of support. As a result, they provide only coarse guidance to a downstream planner and cannot replace detailed feasibility evaluation.

In summary, existing methods either rely on local message-passing GNNs coupled with black-box physics checks, or leverage globally expressive Transformers in settings where physical feasibility is only weakly modelled. None provides a learned, action-conditioned feasibility model that directly predicts the outcome of specific disassembly actions while accounting for both accessibility and global stability, which is the gap we target in this work.

III. FEASIBILITY-AWARE AUTOMATED ASSEMBLY SEQUENCE PLANNING

A. Dataset Generation

Our training data is derived from a large corpus of disassembly graphs (See Fig. 2) generated using the physics-based planning framework from [7]. Each graph edge $e_{t \rightarrow t+1} = (S_t, S_{t+1}, M)$ represents a simulated disassembly attempt, where S_t is the current assembly, $S_{t+1} = S_t \setminus p_j$ is the state after removing part p_j , and $M = \{\text{feasibility, action direction, removed part, stable pose}\}$ contains action metadata. This provides a rich source of supervised data for our action-conditioned learning approach.

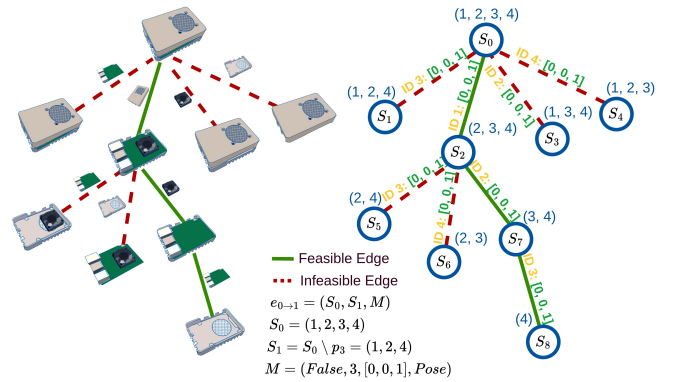


Fig. 2. Disassembly trees used for data generation. *Left*: a disassembly tree visualised with CAD sub-assemblies *Right*: the corresponding abstract tree with node identifiers and action-encoding vectors.

Each attempt is evaluated by two physics-based checks:

Assemblability. We apply a constant translational force to p_j along one of the six principal directions $a_d \in \{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$ and simulate for a fixed horizon. The action is feasible if p_j separates from the assembly without collision; otherwise, it is infeasible (blocked along a_d). Following prior work, restricting actions to these directions provides a conservative proxy for kinematic removability in our benchmark assemblies [6].

Stability. After virtual part removal, the simulation runs for 1000 timesteps while tracking all remaining part positions. The successor state S_{t+1} is deemed unstable if the Euclidean distance between any two parts that were initially in contact changes by more than 0.2 mm, indicating gravitational collapse [7]. An action is labelled feasible ($y = 1$) only if it passes both checks; otherwise, it is infeasible ($y = 0$).

From these annotated graphs, we extract node features for each part: geometric properties (volume, center of mass), spatial context (distance to the assembly centroid), and a 7-dimensional binary action vector $A \in \{0, 1\}^7$ defined as $A = [a_{+X}, a_{-X}, a_{+Y}, a_{-Y}, a_{+Z}, a_{-Z}, a_{\text{blocked}}]$. For each simulated attempt, exactly one of the first six entries is set to 1, indicating the single principal direction that was tested for removal; all other direction entries are 0, making this portion of the encoding strictly one-hot for a given sample. The final component a_{blocked} is set to 1 only when no principal direction is kinematically removable for the queried part. Importantly, a_{blocked} captures kinematic non-removability and is distinct from the overall feasibility label y , which can also be 0 due to post-removal instability even when a direction is kinematically valid. A single attempt encodes one (part, direction) query; a part removable along several directions therefore yields several independent samples, one per tested direction, so the one-hot encoding introduces no ambiguity. Inference (Sec. III-D) mirrors this—every (part, direction) candidate is scored separately, and the highest-scoring feasible one is executed.

B. Dataset Integration and Graph Formulation

Each training sample corresponds to an edge in the disassembly tree (See Fig. 3), connecting a predecessor and a successor state through an action-annotated transition. Our model input is constructed as a part-contact graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ representing the predecessor state S_t , where nodes $\mathcal{V}$ represent parts and edges $\mathcal{E}$ encode physical contact relationships derived from the precomputed global assembly contact graph. The graph construction process involves several critical steps:

Mesh Loading and Transformation: For each part $p_i \in S_t$, we load its mesh and apply the assembly-level transformation to establish correct spatial relationships. However, raw poses in the ASAP dataset can be arbitrarily oriented—some sub-assemblies are tilted or inverted and would collapse if loaded directly into simulation. To ensure physically meaningful inputs, we apply a gravity-alignment preprocessing step: for each assembly, we compute its stable resting pose (via simulation or dataset metadata) and transform the sub-assembly accordingly. This places all training

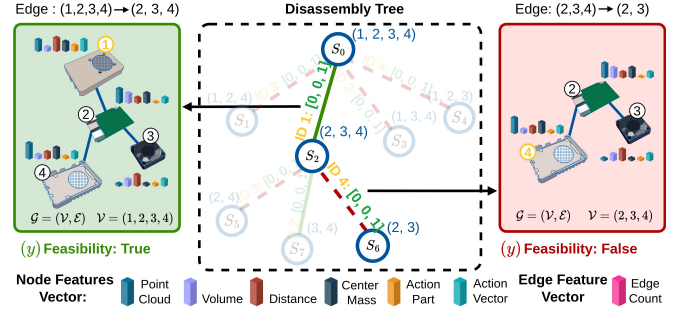


Fig. 3. Integrating Geometry, Constraints, and Feasibility into a contact graph. *Left*: predecessor sub-assembly with feasibility label True. *center*: the disassembly tree. *Right*: successor sub-assembly for the second edge, whose feasibility label is false

samples in a canonical, gravity-aligned reference frame, ensuring geometrically consistent inputs while preserving physical realism.

Feature Extraction: Each part mesh is converted to a point cloud representation through surface sampling, generating N points uniformly distributed across the part’s surface geometry. The point clouds are normalised using the sub-assembly’s bounding box to ensure scale invariance across different assembly sizes. Each node $v_j \in \mathcal{V}$ is enriched with a feature vector containing (See Fig. 3) the point cloud ($\mathbb{R}^{N \times 3}$), part volume, center of mass coordinates ($\mathbb{R}^3$), distance from the part centroid to the sub-assembly centroid, and the 7-dimensional action vector ($\mathbb{R}^7$) defined in Sec. III-A. For the target part, the action vector is populated according to the tested removal direction and blocked flag; other parts receive zero-filled action vectors.

Contact Graph Structure: The contact relationships $\mathcal{E}$ are preserved from the original assembly’s contact graph, maintaining crucial spatial adjacency information that influences disassembly feasibility.

The resulting action-conditioned graph, comprising the structure $\mathcal{G}$, node features, and the binary feasibility label, turns a static structural representation into a dynamic, query-specific input for end-to-end learning. The required configuration (part poses and contact graph) is derived from CAD here, but could equally come from vision-based reconstruction or existing engineering databases in deployment.

C. Feasibility-Aware Sequence Planning Architecture

Our architecture (Fig. 4) assesses the feasibility of disassembly actions through joint reasoning over geometric, physical, and relational assembly information. The network comprises the following four key components:

PointNet (Geometric Encoder): Part geometry is critical for action-specific removal feasibility. Point clouds are unstructured and pose-sensitive, which precludes the use of standard CNNs. We adopt a PointNet encoder for permutation and pose invariance. Five 1D convolutional layers with batch normalisation and ReLU precede global max pooling, yielding a 256-dimensional embedding that encodes local surface features and global shape. This design generalises across assembly geometries without handcrafted priors.

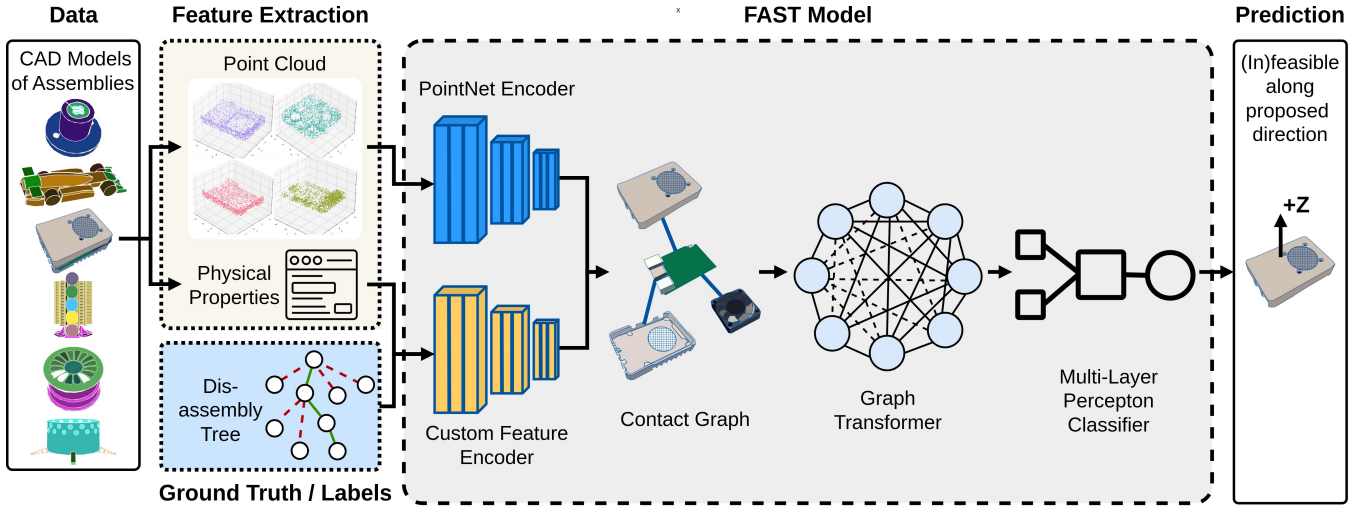


Fig. 4. FAST architecture: with PointNet encoder, Physical Property encoder, Graph Transformer and MLP classifier.

Physical Property Encoder (Contextual Encoder): Geometry alone is insufficient for feasibility; parts must be understood in their physical and spatial context under candidate actions. Our property encoder processes a 13-dimensional feature vector comprising part volume (1D), centre of mass coordinates (3D), distance from part centroid to assembly centroid (1D), an action-target indicator (1D), and the 7-dimensional action encoding, through a three-layer MLP with ReLU, embedding heterogeneous attributes into a 128-dimensional feature space. Action conditioning enables direction-specific reasoning, while spatial features capture stability cues. Ablation studies (Sec. VI-D) validate the contribution of these features, particularly the action vector.

Graph Transformer (Relational Reasoning): Disassembly feasibility depends on inter-part interactions and global stability beyond local neighbourhoods. Unlike message-passing GNNs, which struggle to capture long-range dependencies through iterative aggregation, our Graph Transformer employs global self-attention over all part pairs, regardless of graph distance. The architecture stacks two Graph Transformer layers (8 heads, 256-d, dropout 0.1). Geometric and property features are concatenated into 384-d node representations. Eight attention heads operate on the contact graph to capture diverse relational patterns and compute attention strengths between connected components. This enables reasoning about both local contact constraints and distant stability dependencies, which are essential for complex assemblies.

Graph-Level Classifier (Feasibility Prediction): Node features are aggregated via mean pooling into a global graph representation encoding the sub-assembly state and proposed removal action. A two-layer MLP with ReLU and dropout outputs a feasibility logit. While this indicates directional feasibility, sequential placement is not modelled. We address this by defining a global origin relative to the real-world coordinate system and using the CAD model to specify exact placement coordinates. Dropout ensures generalisation to unseen assemblies during inference.

D. Inference

At inference time, we enumerate all candidate part-action pairs for the current assembly state and evaluate them with FAST. The network assigns a feasibility score to each pair, and we execute the highest-probability feasible action; the process then repeats on the updated state until full disassembly. Crucially, enumerating these pairs is inexpensive: it is a single batched forward pass over at most $6 \times$ the number of remaining parts. FAST thus serves as the *predict* stage of a predict-then-validate loop—the physics simulator validates only the top-ranked action(s) within the evaluation budget. As illustrated in Fig. 5, reversing the resulting disassembly sequence produces a candidate assembly plan. This plan alone is insufficient for execution, since a complete robotic system must also perform object detection, 3D pose estimation, and collision-free trajectory planning. The following section describes the integration of these modules to enable flexible and reliable robotic assembly.

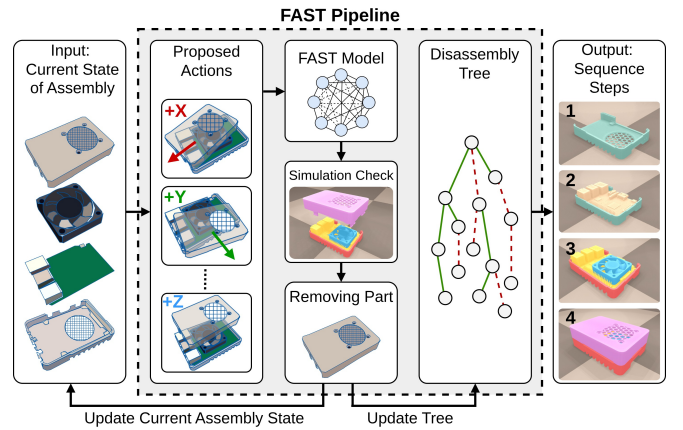


Fig. 5. FAST pipeline: from CAD to a feasible assembly sequence.

IV. END-TO-END ASSEMBLY SYSTEM INTEGRATION

We implement a trajectory planner that exhibits minimal sim-to-real deviation. This module utilises the part assembly

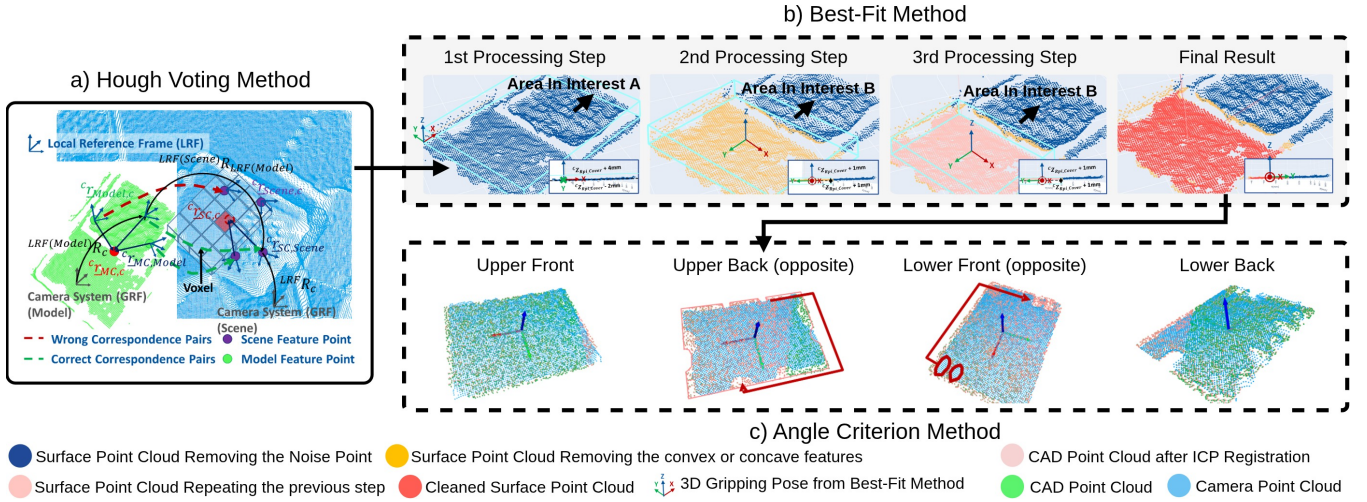


Fig. 6. Object detection and 3D pose estimation pipeline: (a) point cloud registration using Hough Voting; (b) surface refinement via Best-Fit method; (c) registration error correction using Angle Criterion; for a Raspberry Pi platform.

locations generated by the sequence planner while accounting for initial object poses. The processes for identifying these initial poses via object detection and 3D pose estimation in clutter are described below.

A. Vision-Based Object Detection and Pose Estimation

Object detection in cluttered bin environments is challenging due to the vast scale discrepancy between model point clouds (individual parts) and the scene point cloud (containing 1.5-2 million points). The high noise and density of the scene result in extremely low correct pairing probabilities, leading to registration errors and inaccurate 6D pose estimates. To address this, we first apply a Hough voting method [19] to register model point clouds onto the scene. We then enhance this approach with a global registration refinement step to improve robustness in clutter and correct inaccuracies in the initial pose estimates. This integrated pipeline enables reliable detection and pose estimation of diverse objects—including reflective metal parts—despite significant scale differences and high sensor noise.

Fig. 6 illustrates the 3D pose estimation pipeline for the Raspberry Pi cover. The workflow integrates region-based point-cloud acquisition, iterative best-fit plane extraction, iterative closest-point refinement, and an angle-based orientation-disambiguation criterion to achieve millimetre-level precision under challenging conditions. First, a bounding box captures the cover surface, including convex and concave features (Fig. 6a). Points corresponding to surface irregularities are iteratively removed, retaining only stable grasping regions (Fig. 6b-Step 2, 3). This yields four cleaned point-cloud segments: Top Front, Top Back, Bottom Front, and Bottom Back (Fig. 6c). The complex mathematical formulation of best-fit plane extraction is provided in our prior work [20]. It includes singular value decomposition for computing the normal vector and principal axes, along with a detailed angle-based orientation disambiguation procedure. The method was validated on Lego brick pose estimation in that work.

While best-fit plane extraction accurately estimates position, principal axis orientation remains ambiguous due to random initialisation. Direct CAD alignment often fails when point clouds are defective due to occlusion or noise (Fig. 6c Upper Front(opposite) and Lower Front (opposite)). To resolve this, an angle criterion derived from iterative best-fit intermediate results is introduced (Fig. 6c). Validated point clouds undergo final orientation adjustment: if the angles between the Y-axis in the first step and the X-axis in the final step of Fig. 6b are below 5° , axes are swapped to match CAD canonical orientation. The validated pose ${}^C T_{Rpi}$ is transformed to robot base coordinates via the hand-eye calibration matrix ${}^B T_C$, yielding the final grasping pose.

B. Collision-Free Trajectory Planning

Trajectory planning in cluttered, reconfigurable environments involves managing objects of diverse shapes and sizes where obstacles may be moved arbitrarily. To ensure collision-free motion, live point-cloud data is converted into meshes within a digital twin, minimising sim-to-real deviations. A guiding path (Fig. 7a, green path) is generated from predefined poses to direct the planner and limit incorrect node expansions. Sequenced by CAD model frames, this path includes approach points for gripper manoeuvring. Since these frames are generated before collision checking, any CAD inaccuracies may cause collisions, requiring refinement. The process begins by defining start and end poses within the robot's workspace.

The trajectory is refined through a cyclic process starting with 1) Collision Check (Fig. 7a). A dynamic simulation engine performs continuous checking, returning penetration depths and collision vectors, as illustrated in Fig. 8. This diagnostic data, including residual collision pairs, is used to iteratively adjust waypoints opposite to the penetration vectors to ensure a clear path. The complete collision checking pipeline—including environment simplification via basic shape primitives, dynamic collision reporting with penetration depth qualification, and iterative waypoint cor-

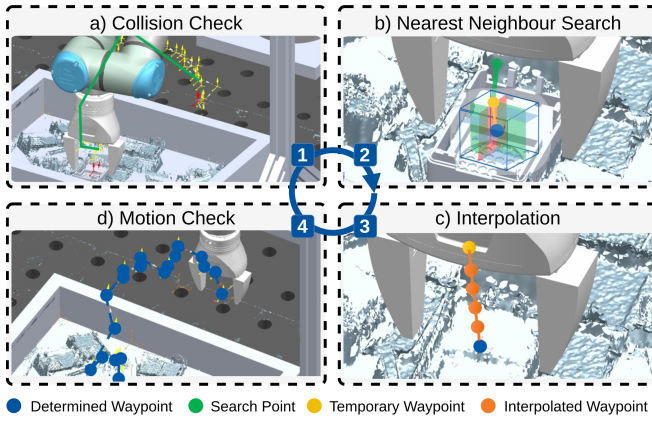


Fig. 7. Iterative trajectory planning workflow involving collision verification, neighbour sampling, and waypoint interpolation.

rection—is detailed in our prior work [20], where we also report quantitative penetration depths across multiple trials (with a maximum of 3.83mm in cluttered bin-picking scenarios). This leads to the 2) Search for Nearest Neighbour (Fig. 7b), where a cubic search space is centred on the current admissible waypoint. In this volume, seven points are uniformly sampled with a specified offset. Symmetric, axis-based partitioning subdivides the space into eight octants, and the nearest neighbour to the target is selected as a candidate for expansion using robust identification techniques.

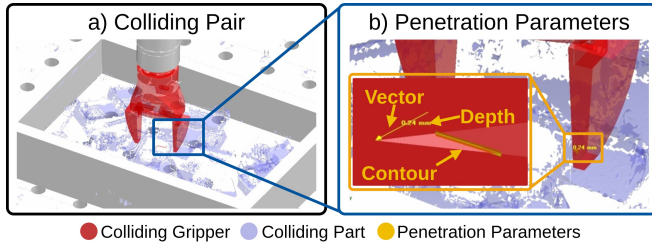


Fig. 8. Visualisation of penetration depth and collision vectors used for iterative trajectory correction.

Following selection, the process moves to 3) Interpolation (Fig. 7c). The trajectory between the start pose and the candidate is linearly interpolated at five equidistant intervals for screening. A coarse collision check is performed at each point to identify a sequence of admissible waypoints. A waypoint is valid only if all five segments are collision-free; at that point, it is stored as the new search origin. Finally, the 4) Motion Check (Fig. 7d) repeats this neighbor search via a k-d tree until the goal is within a defined threshold, finalizing the collision-free trajectory. For our target application—cluttered, reconfigurable industrial settings—we prioritize robustness over optimality. While optimization-based methods may yield smoother paths, sampling-based planning ensures probabilistic completeness, guaranteeing a solution if one exists—a practical advantage over theoretically optimal approaches that often fail in non-convex workspaces.

C. Bridging the Planning-Execution Gap via Passive Mechanical Guidance

To bridge the gap between discrete feasibility checks and continuous contact operations, we introduce a passive

alignment fixture for each base component. The fixture features a funnel-shaped chamber (15° slope) with a ± 3 mm conical capture region that mechanically absorbs positional uncertainty from vision-based pose estimation (Sec. IV-A) and trajectory execution (Sec. IV-B). The robot needs only to place the part within this region; gravity drives self-centring against the sloped walls, and mechanical geometry ensures correct seating. This approach (1) accommodates positioning errors without requiring sub-millimetre precision, and (2) eliminates the need for real-time force control, reducing system complexity and cost. The fixture can be rapidly adapted to new products using standard CAD tools, making it practical for HMLV manufacturing.

V. EXPERIMENTAL SETUP

A. Robotic Assembly System

Our experimental setup is shown in Fig. 1. A Cognex A5120 3D vision system mounted overhead encompasses a workspace containing a bin of unstructured, cluttered components. The system performs object detection to identify a target part—a Raspberry Pi cover—and estimates its 3D pose for grasping. The UR10e robotic arm then executes a pick-and-place operation to acquire the component. Finally, the robot performs the precise assembly, placing the validated cover onto its base unit.

B. Training Details

All training experiments were conducted using a batch size of 128, the Adam optimiser (learning rate 1×10^{-4}), and on-plateau scheduling for adaptive learning-rate adjustment. Training proceeded for up to 150 epochs with an early stopping mechanism (patience=25) to prevent overfitting. The complete training pipeline utilised an NVIDIA GeForce RTX 5090 GPU (32GB of RAM) with a 24-core CPU and 64GB of system memory, achieving convergence in approximately 4 hours.

VI. EVALUATION AND VALIDATION

A. Dataset

Our evaluation uses the 2,146-assembly subset of the Fusion 360 Gallery dataset [21], [22], with parts ranging from 3 to 50+, partitioned into training (1,906 assemblies) and testing (240 assemblies) splits, with balanced complexity distributions (Fig. 9). Unlike prior work validated on simplified block or LEGO assemblies [8], [9], [15], [23], our benchmark features real-world mechanical CAD models with complex interfaces.

From these assemblies, we generated 24,839 unique disassembly edges, each representing an action-feasibility pair. The dataset reflects a realistic distribution: 15,322 feasible and 9,517 infeasible actions. This imbalance is inherent in the disassembly process, as feasible actions transition the assembly to a predecessor state, reducing the action space, whereas infeasible actions leave the state unchanged and are rarely revisited in successful trajectories. Thus, the class distribution mirrors real-world conditions for sequential planning. To mitigate class imbalance during training, we

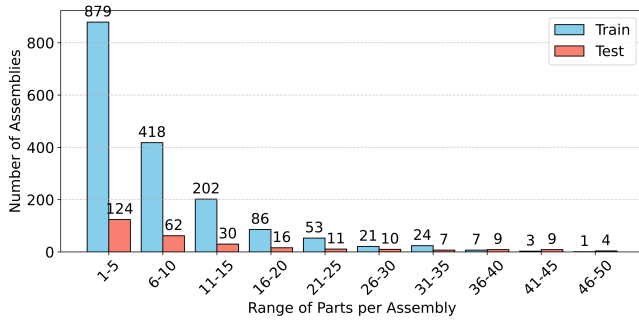


Fig. 9. Number of assemblies by part count (train vs test splits).

apply positive-class weighting to the binary cross-entropy loss.

B. Baseline Method Comparison

We compare against established planners under the same physics-based validation framework (Table I): *Random Permutation* as a lower-bound reference, a *Genetic Algorithm* [12], [13] optimising random permutations via a feasible-step fitness, *Assemble Them All* [6], which searches collision-free sequences but assumes a gravity-free setting without stability, and *ASAP*, the current state of the art, combining heuristic and learning-based part selection with full physics validation. We restrict this quantitative comparison to methods that can be run faithfully under our standardised single-arm, fixed-budget protocol. Recent learning-based planners such as S2A [11] target a different problem setting (subassembly grouping, frequently gravity-free) and do not provide public implementations, which precludes a faithful head-to-head comparison; we therefore position them qualitatively in Sec. II-A. We evaluate sequence-planning output under physics simulation on the 240 held-out test assemblies (disjoint from training). A trial succeeds only when a complete, physically valid sequence is found within the budget.

TABLE I
SUCCESS RATE COMPARISON AGAINST OTHER METHODS

Method	Approach	Success Rate (%) Low Budget	Success Rate (%) High Budget
Baseline	Random Permutation	14.58	27.92
	Genetic Algorithm	14.17	30.83
	Assemble Them All	19.17	30.42
ASAP	Heuristics	51.25	66.67
	Learning	54.58	67.08
FAST (Ours)	Graph Transformer	60.41	80.83

ASAP’s original paper reports high success rates e.g., $>80\%$ (for the case of four parts held simultaneously by auxiliary grippers) by leveraging multiple grippers for stabilisation and extensive simulation, making it effective where parallel holding and high compute budgets are feasible. In contrast, our work targets a more constrained scenario: single-arm assembly without multiple grippers. Under strict simulation budgets (<50 and <400 attempts) reflecting computational efficiency needs, FAST outperforms ASAP by 6%

in the low-budget setting and by 13% in the high-budget setting. FAST achieves this by replacing propose-then-validate with direct, action-conditioned feasibility prediction—better suited to budget-limited, single-arm setups.

C. Planning Efficiency

Fig. 10 compares the average number of physics evaluations required by FAST and ASAP to complete disassembly sequences (high-budget setting). FAST requires substantially fewer evaluations than ASAP across most assembly sizes, scaling more gracefully for complex assemblies (>20 parts). For example, 20-part assemblies require ~ 50 evaluations for FAST vs. ~ 130 for ASAP on average, and 30-part assemblies require ~ 90 vs. ~ 150 . FAST’s std devs are consistently lower, indicating more predictable performance. This efficiency stems from FAST’s direct feasibility prediction, which prunes the search space upfront rather than relying on trial-and-error validation.

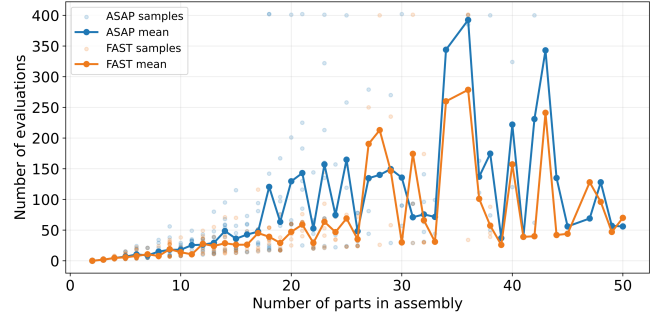


Fig. 10. Method comparison by part count v/s evaluations. Reported counts are physics-simulation evaluations (the dominant cost), not the millisecond-scale network forward passes.

FAST occasionally needs more evaluations than ASAP on a few very large assemblies (e.g., 27–50 parts), where rare geometric edge cases trigger exhaustive exploration; overall, the efficiency gains dominate for the complex assemblies that matter most in practice. Notably, the efficiency gap is negligible for small assemblies but widens markedly for complex ones, where many parts and long contact chains induce long-range stability dependencies. This is consistent with our globally-attending Graph Transformer, which reasons across the entire contact graph rather than through the purely local message passing of the GNN-based baseline, supporting the architectural choice precisely in the regime where it matters most.

D. Ablation Study Analysis

We conduct systematic ablations to validate our feature design choices by quantifying each component’s contribution to feasibility prediction (Table II). The Results validate our action-conditioned formulation: removing the action vector results in the largest accuracy drop (84.47% to 69.43%), confirming that direction-specific information is essential for predicting the feasibility of a specific (part, direction) pair—consistent with our problem definition. Distance to assembly centroid is the second most influential feature

(84.47% to 73.68%), indicating that spatial context (interior vs. exterior) strongly correlates with disassembly feasibility. Part volume and center of mass show smaller impacts. Notably, ablating volume maintains high recall (92.44%), suggesting volumetric information primarily reduces false positives rather than improving overall detection.

TABLE II
ABLATION STUDY OF NODE FEATURES

Ablated Feature	Accuracy(%)	F1-Score(%)	Precision(%)	Recall(%)
Baseline (None)	84.47	90.19	95.04	85.82
Action Vector	69.43↓	75.68	69.54	83.02
Distance	73.68	79.52	71.76	89.15
Volume	75.75	81.37	72.67	92.44
Center Mass	75.00	80.43	72.91	89.67

VII. CONCLUSION

This paper introduces FAST, a feasibility-aware framework for assembly sequence planning that reframes the problem as binary-action prediction. By integrating a Graph Transformer with an action-conditioned data representation, our model directly encodes physical constraints and learns from both successful and failed simulation attempts. Experimental results demonstrate that FAST achieves higher success rates in single-arm, budget-constrained scenarios. Unlike prior systems that assume neatly arranged parts, FAST can be integrated into a complete perception and trajectory-planning pipeline to handle cluttered bin-picking scenarios beyond the scope of existing methods. The explicit modelling of infeasible actions enables robust learning from realistic, imbalanced data distributions while maintaining computational efficiency.

Our current implementation requires watertight CAD models for contact graph generation and stable poses, which limits its use to partial or noisy data. Test assemblies are unseen but share the Fusion 360 distribution; OOD stress-testing remains future work. Real-robot trials reveal practical limitations: CAD interpenetrations cause simulation failures; specular/diffuse reflections on parts and clutter degrade pose estimation; narrow gaps, positioning errors, and limited dexterity cause task failures. These challenges—rather than the sequence planner itself—are the primary inherent bottlenecks in autonomous assembly, and their mitigation is key future work. Future work explores partial point clouds, learned contact models, accelerated physics validation, and generalization without retraining across diverse families.

REFERENCES

- [1] K. Johansen, S. Rao, and M. Ashourpour, "The role of automation in complexities of high-mix in low-volume production—a literature review," *Procedia CIRP*, vol. 104, pp. 1452–1457, 2021.
- [2] Y. Jiang, Z. Huang, B. Yang, and W. Yang, "A review of robotic assembly strategies for the full operation procedure: planning, execution and evaluation," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102366, 2022.
- [3] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," in *Proceedings of the fourteenth annual symposium on Computational geometry*, 1998, pp. 9–18.
- [4] I. Rodriguez, 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*, vol. 4, no. 2, pp. 1416–1423, 2019.
- [5] N. Funk, S. Menzenbach, G. Chaltatzaki, and J. Peters, "Graph-based reinforcement learning meets mixed integer programs: An application to 3d robot assembly discovery," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 215–10 222.
- [6] 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)*, vol. 41, no. 6, pp. 1–11, 2022.
- [7] 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)*. IEEE, 2024, pp. 4380–4386.
- [8] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham, "Can robots assemble an ikea chair?" *Science Robotics*, vol. 3, no. 17, p. eaat6385, 2018.
- [9] 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)*. IEEE, 2020, pp. 9088–9095.
- [10] R. Liu, R. Chen, A. Abuduweili, and C. Liu, "Proactive human-robot co-assembly: Leveraging human intention prediction and robust safe control," in *2023 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2023, pp. 339–345.
- [11] C. Shu, A. Kim, and S. Park, "Subassembly to full assembly: Effective assembly sequence planning through graph-based reinforcement learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 946–14 952.
- [12] E. Kongar and S. M. Gupta, "Disassembly sequencing using genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 30, no. 5, pp. 497–506, 2006.
- [13] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang, "A block-based genetic algorithm for disassembly sequence planning," *Expert Systems with Applications*, vol. 96, pp. 492–505, 2018.
- [14] R. Liu, A. Chen, W. Zhao, and C. Liu, "Physics-aware combinatorial assembly sequence planning using data-free action masking," *IEEE Robotics and Automation Letters*, 2025.
- [15] L. Ma, J. Gong, H. Xu, H. Chen, H. Zhao, W. Huang, and G. Zhou, "Planning assembly sequence with graph transformer," *arXiv preprint arXiv:2210.05236*, 2022.
- [16] B. Xu, S. Zheng, and Q. Jin, "Spaformer: Sequential 3d part assembly with transformers," in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 1317–1327.
- [17] J. Ge, M. Zhou, and C.-W. Fu, "Learn to create simple lego micro buildings," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–13, 2024.
- [18] A. Cebulla, T. Asfour, and T. Kröger, "Speeding up assembly sequence planning through learning removability probabilities," in *ICRA, 2023*, pp. 12 388–12 394.
- [19] F. Tombari and L. Di Stefano, "Object recognition in 3d scenes with occlusions and clutter by hough voting," in *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*. IEEE, 2010, pp. 349–355.
- [20] X. Xu, A. Bashir, J. Bhagiya, and R. Müller, "Collision-free trajectory planning in cluttered environments for efficient bin picking," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 3937–3944.
- [21] K. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama *et al.*, "Joinable: Learning bottom-up assembly of parametric cad joints," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 15 849–15 860.
- [22] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot, "Content-based multi-criteria similarity assessment of cad assembly models," *Computers in Industry*, vol. 112, p. 103111, 2019.
- [23] A. Pun, K. Deng, R. Liu, D. Ramanan, C. Liu, and J.-Y. Zhu, "Generating physically stable and buildable brick structures from text," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 14 798–14 809.