2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Leveraging Two Robotic Arms for Tight Assembly Performance Gains

Dror Livnat† Yuval Lavi† Michael M. Bilevich† Dan Halperin†

***Abstract*—We provide a novel end-to-end framework for the execution of an assembly operation by two robotic arms, given the digital CAD models of the parts and their desired relative placement in their assembled state. We analyze and demonstrate the advantages of using two robotic arms simultaneously in tight assembly operations, compared to single-arm systems. Our method is implemented in both simulation and using physical robots. It provides theoretical guarantees on execution time and trajectory accuracy, supported by empirical evidence. In particular, we show that coordinated movement of two arms reduces average execution time by more than 50% compared to using a single arm only, produces higher-quality trajectories, and accelerates the search for valid robot placements. Furthermore, we establish bounds on the required dimensions of the robotic cell.**

Our open source software together with real-life video demonstrations are available in our project page.

I. Introduction

In modern manufacturing, assembly consumes about 50% of the total production time [1]–[4]. Therefore, approaches that can significantly shorten assembly time have a strong impact on the industry. In this work, we present such an approach: a framework that translates CAD models into robotic arm motion plans, substantially accelerating assembly processes (Fig. 1). We note that the present study focuses on one complete industrial assembly operation as a representative case, rather than an entire full-product assembly sequence, in order to isolate and address the fundamental planning challenges.

The growing demand for mass customization requires flexibility in production, often achieved during product assembly [5]. Consequently, assembly planning and robotic assembly have become prominent fields of research [1], [6].

Industrial robots—especially in assembly—are widely acknowledged as central to Industry 4.0, driving improvements in flexibility, throughput, and quality [7]. However, their widespread deployment is still hindered by the need for expert programming and complex system setup, which remains a significant bottleneck to agility and scalability [8], [9].

In a recent work [10] we presented a framework for converting a digital design of assembly parts into a single robotic arm trajectory plan, together with the desired trajectory placement.

That work addressed three main challenges. First was the well studied free-flying objects motion planning [11]–[25], which to date is still challenging in tight environments. Second was coping with the Continuous Path-Wise Inverse Kinematics (*CPW-IK*), which converted a continuous assembly trajectory in the workspace into a continuous trajectory in the joint-space, overcoming challenges, such as singularities, different IK branches, and collision avoidance. The IK itself is also a well studied challenge [26]–[35]. However, no existing solution could handle the continuous task for UR-like arms efficiently enough to support trajectory-placement search. Hence, a dynamic programming based solution coupled with analytic IK [28] was presented. Third was the challenge of *trajectory placement*. This is dual to the well-studied *robot placement* problem [36], in which feasible base locations are identified for each desired pose, or set of poses, of the robot end effector. Further work was done on mapping and testing robot placement for multiple isolated configurations [36] in order for the robot to be able to complete a path. However, the challenge solved in [10] was to place a full continuous trajectory in a complex $6D$ environment.

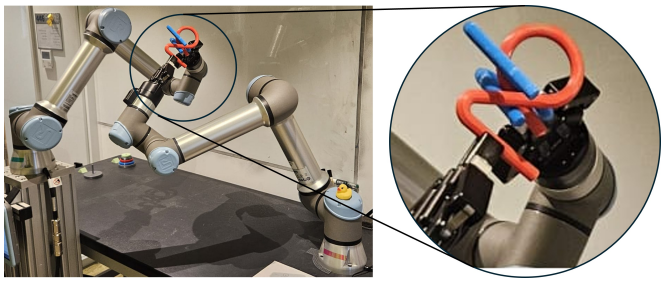

Fig. 1: Coordinated assembly using two robotic arms.

Other works have provided a full cycle assembly planning for simpler tasks using a single robotic arm [10], [37], [38], or construct motion plans, such that the transformation between its end effectors remains fixed [39]. Using two arms can accelerate execution of tabletop rearrangement task as shown in [40]. However, we are not aware of any prior method that plans complex workspace assembly trajectories for two robotic arms.

In this work, we introduce such a framework, complemented by theoretical guarantees and experimental validation. Our results demonstrate clear benefits over single-arm systems: substantially shorter execution times, increased flexibility, and improved accuracy.

This paper makes the following contributions:

- We introduce a new framework and two algorithms, **CPW-IK** and **GPW-IK**, that generate coordinated two-arm assembly plans directly from CAD models.
- We present a first theoretical analysis proving that

†Blavatnik School of Computer Science and Artificial Intelligence, Tel-Aviv University, Israel. This work has been supported in part by the Israel Science Foundation (grant no 3598/25), by the Blavatnik Computer Science Research Fund, and by the Shlomo Shmelzer Institute for Smart Transportation at Tel Aviv University.

average assembly operation time, using the new GPW-IK algorithm, is reduced by over 50%.
- We provide extensive experimental validation demonstrating significant gains in assembly time, trajectory accuracy, and algorithm runtime relative to single-arm baselines.
- We validate the approach using real robotic hardware, demonstrating practical feasibility.

Our open source software and real-life demonstrations are available on our project website: https://www.cgl.cs.tau.ac.il/projects/advantages-of-using-two-robotic-arms-for-tight-assemblies/.

II. Preliminaries and Problem Statement

In this section we summarize tools from [10] needed for tight motion planning of free-flying objects, as well as motion planning using a single robotic arm. We then state the problem that we study in this work.

A. Notation

We briefly summarize the notation used throughout the paper. Configurations in the *workspace* (i.e., rigid-body poses in $\mathrm{SE}(3)$) are denoted with a bar, e.g., $\bar{q} \in \mathrm{SE}(3)$. Joint configurations in the *configuration space* of a robotic arm are written without a bar, e.g., $c \in \mathbb{R}^6$ for a 6-DoF manipulator. We denote by $C_{\text{free}} \subseteq \mathbb{R}^6$ the collision-free subset of the joint configuration space, and $\bar{C}_{free} \in \mathrm{SE}(3)$ its workspace equivalent. Paths are written as continuous functions $\gamma : [0,1] \to C_{\text{free}}$ in joint space and $\bar{\gamma} : [0,1] \to \mathrm{SE}(3)$ in workspace. When considering two arms, we use $C^2 = C^1 \times C^2$ to denote the combined configuration space.

B. Tight Motion Planning of Free-Flying Objects

The input to this phase is a free-flying assembly problem. Specifically, we are given digital models of two free-flying bodies, B_1 and B_2, together with start and goal configurations $\bar{q}_{\text{start}}$ and $\bar{q}_{\text{goal}}$ for one of the bodies. The other body is assumed to remain fixed at the origin. We deploy an appropriate motion planning algorithm, e.g., as described in [24], [25], [41]. Specifically, in this work we use the TR-RRT algorithm from [25] as it best performs on the dataset presented in [24], which is also in use in the current work.

Most sampling-based algorithms, including those in [24], [25], [41] suffer from two undesired properties. First, they usually allow for a small penetration between the sub-assemblies. This works well in simulation, but is challenging to realize in the physical world. Second, the output trajectories might be, as is often the case with sampling-based plans, unnecessarily long and jagged. We therefore follow the free-flying motion planning algorithm with a post processing phase. This includes the *retract* function from [25], to remove the penetrations, followed by a smoothing function as described in [10].

The output of this phase is a collision free, piecewise linear path $\bar{\gamma}$, from $\bar{q}_{\text{start}} \in \bar{\mathcal{C}}_{\text{free}}$ to $\bar{q}_{\text{goal}} \in \bar{\mathcal{C}}_{\text{free}}$, i.e., a path $\bar{\gamma} : [0,1] \to \bar{\mathcal{C}}_{\text{free}}$ such that $\bar{\gamma}(0) = \bar{q}_{\text{start}}$ and $\bar{\gamma}(1) = \bar{q}_{\text{goal}}$. The full sequence is

$$\bar{q}_{\text{start}} = \bar{q}_0,\ \bar{q}_1,\ \ldots,\ \bar{q}_{N-1},\ \bar{q}_N = \bar{q}_{\text{goal}}, \tag{1}$$

and $\bar{\gamma} \in \bar{\mathcal{C}}_{\text{free}}$ consists of the segments between each pair of consecutive configurations in the sequence.

C. Continuous Path-wise IK for a Single Robotic Arm

In this section, we summarize the method presented in [10]. One major difficulty of transferring a trajectory from $\mathrm{SE}(3)$ to the joint-angle space is finding a consistent branch of inverse-kinematics [30]. For each end-effector pose in $\mathrm{SE}(3)$, the six-DoF UR-like robotic arm has up to eight possible different inverse-kinematics solutions, which can be analytically computed.

For each pose in the trajectory of a free-flying object a fixed pose of the respective end-effector is induced based on a predefined grasping. For that end-effector desired pose, all possible IK solutions are calculated. Then, a layered Directed Acyclic Graph (DAG) $G = (V, E)$ is constructed, where layer L_k contains the feasible IK solutions of pose k in the original trajectory. A node $a \in L_k$ is connected with a directed edge e to node $b \in L_{k+1}$ if and only if their L^1 distance in the joint-angle space is smaller than some predefined threshold δ. The weight of the edge is that same L^1 distance $w(e) = ||a - b||_{L^1}$. Two dummy nodes s, t are added to the graph, and are connected with zero-weight edges to the nodes of layers L_1 and L_N respectively, with N being the number of poses in the original trajectory. Denote by $\Pi_G(s,t)$ all paths from s to t in the graph G.

The algorithm finds a path γ whose heaviest edge has the smallest weight. That is, it chooses γ:

$$\gamma = \mathrm{argmin}_{\rho \in \Pi_G(s,t)} \left\{ \max_{e \in \rho} w(e) \right\}. \tag{2}$$

The *error* of the trajectory γ is defined as follows: Let $p \in \partial A$ be a point on the boundary of the sub-assembly manipulated by the robotic arm. Let $\gamma_p, \bar{\gamma}_p : [0,1] \to \mathbb{R}^3$ be the trajectories of p after following the paths γ and $\bar{\gamma}$, respectively. The one-sided Hausdorff distance between γ_p and $\bar{\gamma}_p$ is $d(\gamma_p, \bar{\gamma}_p) = \max_{x \in \gamma_p} \min_{y \in \bar{\gamma}_p} ||x - y||$. Then the error of the path γ is the largest one-sided Hausdorff distance of any point $p \in \partial A$:

$$\mathrm{Err}(\gamma) = \max_{p \in \partial A} d(\gamma_p,\ \bar{\gamma}_p)\,. \tag{3}$$

Our goal is to bound this error by some arbitrarily small ε.

Denote by D the maximal distance of a point in the moving system (the robot and the dynamic body attached to it) from the robot's base. Then, it is shown in [10] that limiting δ_γ, the maximal weight of an edge along the trajectory, $\delta_\gamma \leqslant \frac{\varepsilon}{D}$, yields the desired error bound $\mathrm{Err}(\gamma) \leqslant \varepsilon$.

D. Trajectory Placement for a Single Robotic Arm

For a single robotic arm, choosing merely the initial pose of the dynamic sub-assembly B_2, which is referred as *placement*, determines the entire placement of the trajectory. Note that not all placements have a valid corresponding trajectory in joint-angle space. Hence, for a single robotic arm, we begin by randomly sampling a six-dimensional starting pose, and run the CPW-IK algorithm. If successful, a valid placement for the trajectory is found; otherwise, we sample a new initial pose and repeat the process.

E. Problem Statement

The input to the framework consists of CAD models of two rigid bodies to be assembled, together with desired start and goal relative poses in $\mathrm{SE}(3)$. The output is detailed instructions, in joint-angle space, of how each of two robotic arms should move in order to perform the assembly. This includes placement of the pieces at the beginning of the assembly, grasping poses, and a *time-parameterized trajectory* of each robotic arm. Formally, we start with two rigid bodies $B_1, B_2 \subseteq \mathbb{R}^3$, which we wish to assemble. We assume that the bodies are placed in a pose aligned with their grasp by a robotic arm. In the CAD software however, they may be posed differently, with offsets of $\bar{q}_{B_1}, \bar{q}_{B_2} \in \mathrm{SE}(3)$ for B_1, B_2, respectively. Hence, when planning for free-flying objects, we can assume that B_1 is fixed at the pose $\bar{q}_{B_1}$, and that B_2 is a dynamic object that we wish to move from $\bar{q}_{B_2}$ to $\bar{q}_{\text{goal}} \cdot \bar{q}_{B_2}$.

The configuration space (C-space) of a single UR-like robotic arm is $\mathcal{C} = [0, 4\pi)^6 \subseteq \mathbb{R}^6$. The forward kinematics function $f : \mathcal{C} \to \mathrm{SE}(3)$ maps a configuration $c \in C$ to its corresponding workspace pose $\bar{q} = f(c)$ of the end-effector.

When two robotic arms R_1, R_2 are in play, the combined configuration space is the product $\mathcal{C}^2 = \mathcal{C} \times \mathcal{C} \subset \mathbb{R}^{12}$. That is $c^2 = (c^{(1)}, c^{(2)}) \in \mathcal{C}^2$ is a pose such that the arm R_1 is at configuration $c^{(1)}$ and arm R_2 at configuration $c^{(2)}$.

We say that $c^2 \in \mathcal{C}^2_{forbid} \subseteq \mathcal{C}^2$ if any two of R_1, R_2, B_1, B_2 are in intersection among themselves or with an obstacle.

We define the *free region* as all configurations that yield no intersection of the interiors of the robots, the objects, and the environment obstacles:

$$\mathcal{C}^2_{free} = \left(\mathcal{C}^2 \backslash \mathcal{C}^2_{forbid}\right). \tag{4}$$

We are now ready to define our problem.

Problem Statement: We are given CAD models of two rigid bodies $B_1, B_2 \subseteq \mathbb{R}^3$, their initial configurations $\bar{q}_{B_1}, \bar{q}_{B_2} \in \mathrm{SE}(3)$, and a desired goal pose $\bar{q}_{\text{goal}}$ for body B_2. The objective is to compute a collision-free motion path $\gamma = (\gamma^{(1)}, \gamma^{(2)}) : [0,1] \to \mathcal{C}^2_{\text{free}}$ that satisfies the following conditions:

$$f\left(\gamma^{(1)}(0)\right)^{-1} \cdot f\left(\gamma^{(2)}(0)\right) = (\bar{q}_{B_1})^{-1} \cdot \bar{q}_{B_2}\,, \tag{5}$$

$$f\left(\gamma^{(1)}(1)\right)^{-1} \cdot f\left(\gamma^{(2)}(1)\right) = (\bar{q}_{B_1})^{-1} \cdot (\bar{q}_{\text{goal}} \cdot \bar{q}_{B_2})\,. \tag{6}$$

These constraints ensure that the relative configuration of B_2 in the frame of B_1 matches the given initial conditions at $t = 0$ (equation 5) and the desired goal configuration at $t = 1$ (equation 6).

We note that a choice for such a path γ also incorporates a choice for the placement of the rigid bodies in the workspace. That is, γ may be any valid trajectory in joint-angle space that starts with assembled pieces and ends with dis-assembled pieces.

In this work we restrict ourselves to paths that are piecewise linear. A path

$$\gamma = \{c^2_{\text{start}} = c^2_0,\ c^2_1,\ \ldots,\ c^2_{N-1},\ c^2_N = c^2_{\text{goal}}\}, \tag{7}$$

is valid, if the segment between each pair of consecutive configurations in the sequence is contained in C^2_{free}.

Within this general definition of γ, we explicitly implement three types of trajectories. First is a single arm trajectory as presented in [10], where arm R_1 keeps body B_1 static, and arm R_2 does all the work with B_2. The second and the third, which are defined in the next section, are two different algorithms providing *coordinated two-arm plan*, where at each step both of the arms may move.

III. Method

In this section we show how the single arm CPW-IK method presented in section II-C can be extended to the case of two arms, using two different approaches. The input to this phase is the output of the free-flying objects motion plan, which is a path $\bar{\gamma} : [0,1] \to \bar{\mathcal{C}}_{\text{free}}$ such that $\bar{\gamma}(0) = \bar{q}_{\text{start}}$ and $\bar{\gamma}(1) = \bar{q}_{\text{goal}}$, as discussed in section II-B.

To develop algorithms using two robotic arms, we refine the definition of a *segment* as follows. For the two-arm execution phase, segment i is anchored by the *absolute* world poses of the two bodies at its start,

$$\sigma_i = (\bar{p}_{1,i},\ \bar{p}_{2,i}) \in \mathrm{SE}(3) \times \mathrm{SE}(3),$$

together with the next *relative* target pose $\bar{q}_{i+1} \in \mathrm{SE}(3)$ induced by the free-flying path. Our relative-pose convention is

$$\bar{q} = \bar{p}_1^{-1} \bar{p}_2,$$

such that the segment begins at $\bar{q}_i = \bar{p}_{1,i}^{-1} \bar{p}_{2,i}$ and ends at any pair of absolute poses $(\bar{p}_{1,i+1}, \bar{p}_{2,i+1})$ satisfying the constraint

$$\bar{p}_{1,i+1}^{-1} \bar{p}_{2,i+1} = \bar{q}_{i+1}.$$

This freedom of choosing the absolute end pose of B_1 (and hence of B_2) while meeting the required relative target is the key ingredient that allows the coordinated two-arm execution-time gains. In both algorithms we go over the free-flying trajectory $\bar{\gamma}$ step by step, and use the end pose from segment i as the starting pose for segment $i+1$. We aim to find a corresponding collision-free path $\gamma : [0,1] \to \mathcal{C}^2$ in the two robotic arms joint space.

Execution-time model. For each motion segment, we assume synchronized joint execution: the joint(s) with maximal angular displacement move at a fixed maximal velocity, and the remaining joints move at constant lower velocities so all joints complete the segment together. Hence, segment time is proportional to the maximum joint angular displacement in that segment.

A. CPW-IK for Coordinated Two-Arms

First, we sample a random placement for object B_1. That is, we sample a random $\bar{q}_{rand} \in \mathrm{SE}(3)$.

For each step $\bar{q}_i \mapsto \bar{q}_{i+1}$ in the desired trajectory $\bar{\gamma}$ we do as follows:

- Calculate the midpoint between $\bar{q}_i$ and $\bar{q}_{i+1}$.

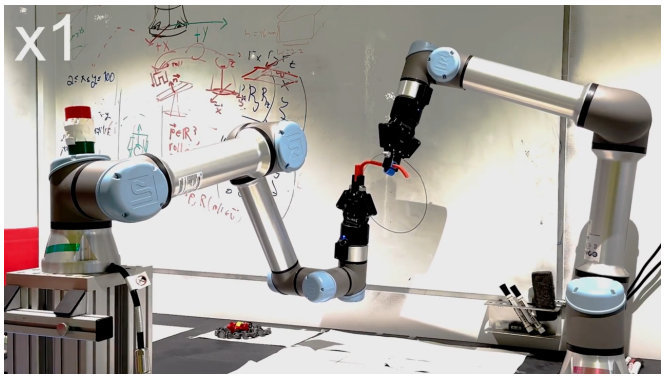

Fig. 2: Coordinated assembly of industrial assembly number 16505.

- Find all IK solutions for that midpoint pose, and add those solutions as a layer in the dynamic DAG, for the object B_2.
- Find the corresponding pose for part B_1 in $\mathrm{SE}(3)$ by multiplying the new pose of B_2 by the transformation matrix from that same pose in $\bar{\gamma}$ and $\bar{q}_{B_1}$. Find all possible IK solutions for that pose and add the layer to the DAG, for the rigid part B_1.

Each edge in these layers, corresponds to simultaneous movement of the robotic arms R_1 and R_2 towards that midpoint. After building the DAG, we use the same dynamic programming presented in Section II-C to select a path minimizing the one-sided Hausdorff distance. That is $d_H(\{f(\gamma^{(1)}(t))^{-1} f(\gamma^{(2)}(t)) : t \in [0,1]\}, \{q_{B_1}^{-1}\bar{\gamma}(t) : t \in [0,1]\})$, the distance between the realized trajectory of B_2 in the frame of B_1 obtained via forward kinematics and the desired trajectory of B_2 specified in the input. Alternatively, we minimize the makespan of the path.

B. Greedy Path-Wise IK for Coordinated Two-Arms

We now present the Greedy Path-Wise Inverse Kinematics (GPW-IK) algorithm, our new coordinated planning algorithm, which is the first to guarantee more than 50% reduction in per-average-segment execution time compared to independent arm motions. Throughout this section, $\bar{q}_i$ denotes the *relative configuration of B_2 in the frame of B_1*, with the convention $\bar{q}_i = \bar{p}_{1,i}^{-1}\bar{p}_{2,i}$. Thus, a segment is defined by the relative update $\bar{q}_i \mapsto \bar{q}_{i+1}$, while the absolute end poses $(\bar{p}_{1,i+1}, \bar{p}_{2,i+1})$ are free to vary subject to $\bar{p}_{1,i+1}^{-1}\bar{p}_{2,i+1} = \bar{q}_{i+1}$. Although segments are indexed by relative configurations $\bar{q}_i \to \bar{q}_{i+1}$, a segment realization is defined by a transition between absolute world poses $(\bar{p}_{1,i}, \bar{p}_{2,i}) \to (\bar{p}_{1,i+1}, \bar{p}_{2,i+1})$ that satisfies $\bar{p}_{1,i}^{-1}\bar{p}_{2,i} = \bar{q}_i$ and $\bar{p}_{1,i+1}^{-1}\bar{p}_{2,i+1} = \bar{q}_{i+1}$.

The GPW-IK algorithm synchronizes the execution of the two arms so that they complete the work on each segment simultaneously. Instead of splitting evenly as in CPW-IK, GPW-IK computes a per-segment split that minimizes the common completion time.

a) Per-segment rule: Let t_1 denote the time required for R_1 to execute the entire segment alone (keeping R_2 static), and let t_2 be the analogous time for R_2. GPW-IK chooses a split

$$x = \frac{t_2}{t_1 + t_2}, \qquad 1 - x = \frac{t_1}{t_1 + t_2},$$

so that both arms complete their partial motions in the same synchronized time

$$T = \frac{t_1 t_2}{t_1 + t_2}.$$

b) Toy example: Suppose $t_1 = 6\,\mathrm{s}$ and $t_2 = 4\,\mathrm{s}$. Then $x = 0.4$ and $1 - x = 0.6$, yielding a synchronized execution time $T = 2.4\,\mathrm{s}$. By comparison, CPW-IK's even split would take $\max(0.5 \cdot 6, 0.5 \cdot 4) = 3\,\mathrm{s}$, so GPW-IK achieves a shorter makespan.

c) Algorithm (per segment): For each segment $\bar{q}_i \mapsto \bar{q}_{i+1}$, GPW-IK performs:

1) Take into account all possible (up to 8) IK solutions for each arm, and select the one that minimizes the time for that step for each arm, if it were to perform the segment alone. Compute the single-arm execution times t_1 and t_2 as above.
2) Compute the synchronization fractions $x = \frac{t_2}{t_1 + t_2}$ and $1 - x = \frac{t_1}{t_1 + t_2}$.
3) **Partial move of R_2:** move each joint of R_2 by fraction $1 - x$ of the way toward the configuration that would realize $\bar{q}_{i+1}$ if R_1 stayed fixed.
4) **Placement of B_1:** place B_1 so that B_2 is in the required relative configuration $\bar{q}_{i+1}$. More formally, if $p_i \in \mathrm{SE}(3)$ is the pose of B_i in world coordinates, move B_1 to

$$\bar{p}_1^{\text{target}} = \bar{p}_2^{(\text{partial})} (\bar{q}_{i+1})^{-1}.$$

5) **Move of R_1:** calculate the IK for $\bar{p}_1^{\text{target}}$ and move R_1 accordingly.
6) Proceed to the next segment.

d) Comparison to CPW-IK: CPW-IK enforces an equal (50/50) split, which is only optimal if $t_1 = t_2$. GPW-IK adapts the split per segment: if $t_1 \neq t_2$ it assigns more of the motion to the faster arm and strictly improves the makespan. If $t_1 = t_2$, the two methods coincide.

IV. Analysis and Guarantees

A. GPW-IK Analysis

We analyze the execution time of GPW-IK under the standard joint-velocity model used throughout this paper: for each segment, the joint requiring the largest angular displacement moves at its maximal constant speed, while all other joints move at equal or lower constant speeds. Consequently, the execution time of a segment is proportional to the maximum joint angle traversed along that segment. Under this model, we compare the synchronized two-arm execution of a segment to its single-arm counterpart.

In this section, $\bar{q}_i$ denotes the *relative configuration of B_2 in the frame of B_1*. A segment is therefore defined by a change $\bar{q}_i \mapsto \bar{q}_{i+1}$.

In order to estimate the expected makespan improvement that GPW-IK provides over the usage of a single arm, we look at all possible short segments in the workspace, and compare the time it takes two robotic arms using GPW-IK to complete the motion along such a segment, to the time it would have taken a single arm.

Consider such a short segment $s := \bar{q}_i \mapsto \bar{q}_{i+1}$, and consider two identical robotic arms in a shared workspace. For a given motion segment, or step:

- Arm R_1 requires $t_1 > 0$ time to execute the entire motion.
- Arm R_2 requires $t_2 > 0$ time to execute the entire motion.

In GPW-IK we let the arms execute complementary fractions of the motion such that they finish simultaneously.

Let x be a fraction of the motion performed by arm R_1. That is, if $(r_1, r_2, ..., r_6) = c_{i+1}^{(1)} - c_i^{(1)}$ is the set of rotations required by R_1 in order to move the assembly from $\bar{q}_i$ to $\bar{q}_{i+1}$, then moving a fraction x is making the step $x \cdot (r_1, r_2, ..., r_6)$. Then the times of arms R_1 and R_2 are xt_1 and $(1-x)t_2$ respectively. For simultaneous completion:

$$xt_1 = (1-x)t_2,$$

which gives:

$$x = \frac{t_2}{t_1 + t_2} \text{ and } 1 - x = \frac{t_1}{t_1 + t_2}.$$

The common completion time is:

$$T_{\textit{two-arms}} = xt_1 = (1-x)t_2 = \frac{t_1 t_2}{t_1 + t_2}$$

By symmetry over the workspace, there exists an equivalent segment where the times are swapped ($t_1 \longleftrightarrow t_2$). For that segment, the optimal completion time is the same:

$$T_{\textit{two-arms}} = \frac{t_1 t_2}{t_1 + t_2}.$$

Thus, the time for completion of both segments combined in a two-arm setting:

$$T_{\textit{total, two-arms}} = 2 \cdot \frac{t_1 t_2}{t_1 + t_2}.$$

In a single arm setting, the total time it takes a single arm R_1 to execute both segments alone is:

$$T_{\textit{total, single}} = t_1 + t_2.$$

The time ratio (two-arms setting vs. single-arm setting) for completion of both segments is then:

$$\rho = \frac{T_{\textit{total, two-arms}}}{T_{\textit{total, single}}} = \frac{2 \cdot \frac{t_1 t_2}{t_1 + t_2}}{t_1 + t_2} = \frac{2 t_1 t_2}{(t_1 + t_2)^2}.$$

When $t_1 = t_2$, we have:

$$\rho = \frac{2t_1^2}{(2t_1)^2} = \frac{1}{2}.$$

This is the maximum possible ratio. For unequal times $t_1 \neq t_2 > 0$ we have from the Arithmetic Mean-Geometric Mean inequality:

$$\frac{t_1 + t_2}{2} \geqslant \sqrt{t_1 t_2}.$$

Squaring both sides and rearranging yields an upper bound for ρ:

$$\rho = \frac{2 t_1 t_2}{(t_1 + t_2)^2} \leqslant \frac{1}{2}.$$

This analysis shows that, on average, the execution time for a short segment is reduced by more than 50% compared to single-arm execution. This is also true for *average* collections of such segments. Although this gives hope to find trajectories for two robotic arms that save more than 50% of the time, we do not claim that it guarantees that the *optimal* two-arm solution is shorter by more than 50% than the *optimal* single arm solution, as they do not consist of the same collection of segments. We therefore ran a large series of experiments to assess the practical strength of the algorithm.

B. Path Accuracy

Here we briefly outline how the guarantee for path accuracy developed in [10] and reviewed in II-C can be extended to the two-arm case. Let $p \in \partial A$ be any point on the boundary of an assembly part. Consider a segment $(c_i^2,\, c_{i+1}^2)$. Its endpoints, c_i^2 and c_{i+1}^2 serve as anchor points where, by construction, the error, or the distance between any point p and where it should have been according to the free-flying trajectory $\bar{\gamma}$ is zero (see step 4 in the GPW-IK algorithm).

We begin by redefining D: instead of denoting the distance from the base of an arm to the farthest point on the single assembly piece it manipulates, D now represents the distance from the base of an arm to the farthest point on any of the assembly pieces. Recall that the resolution of the input free-flying trajectory $\bar{\gamma}$ is such that the maximal angle required to traverse a segment $q_i \mapsto q_{i+i}$ is some δ. In GPW-IK, the sum of the maximal angles α_1, α_2 traversed by R_1,R_2 respectively along a single segment is therefore smaller than that δ, $\alpha_1 + \alpha_2 < \delta$. For the first half of the motion within a segment, we therefore have $\alpha_1 + \alpha_2 \leqslant \frac{\delta}{2}$. In this setting, the maximum displacement of a point $p \in \partial A$ in $\mathbb{R}^3$ is bounded by $\frac{\alpha_2}{2} \cdot D$. At the same time, its reference point p', initially coincident with p and displaced by arm R_1, moves by at most $\frac{\alpha_1}{2} \cdot D$. Hence, the maximal discrepancy between p and p' is bounded by $(\frac{\alpha_1}{2} + \frac{\alpha_2}{2}) \cdot D \leqslant \frac{\delta}{2} \cdot D$. By symmetry, and since c_{i+1}^2 is also an anchor point where the distance between any point p and its reference point p' is zero, the same reasoning applies throughout the entire segment. Consequently, imposing the condition $\delta \leqslant \frac{\epsilon}{D}$, as in the single-arm case, ensures that the maximal one-sided Hausdorff distance of any point p is $\mathrm{ERR}(\delta) \leqslant \frac{\epsilon}{2}$, i.e., exactly half the error guarantee of the single-arm scenario in [10].

C. Robotic Cell Size Guarantee

In this section, we provide guarantees regarding the space that the two-arm robotic cell will occupy. This is based solely on the components involved and the positions of the robotic arms, regardless of the specific trajectory and placement.

Understanding the dimensions of the robotic cell is crucial when deploying robotic arms in factories and workshops. It ensures safety, facilitates collision avoidance, and optimizes the use of floor space.

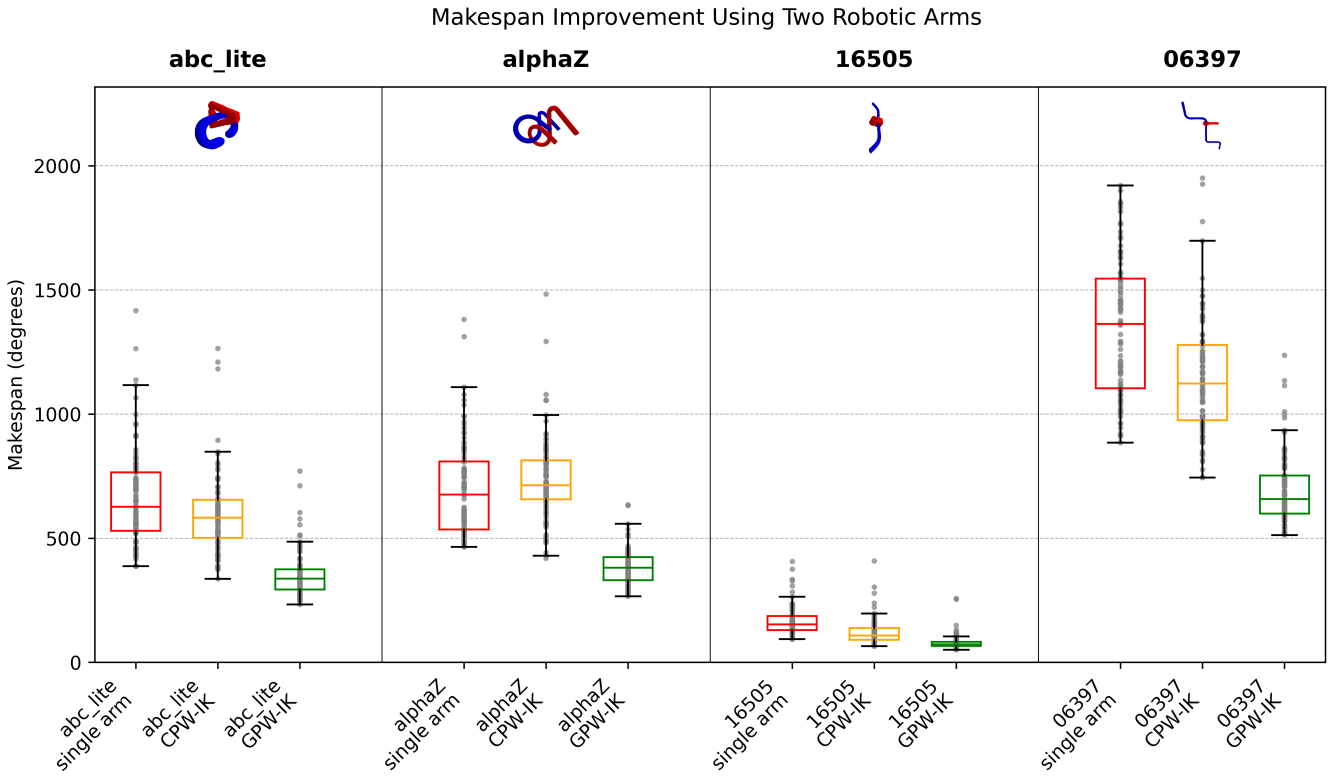

Fig. 3: Makespan (total execution time) for each assembly and each IK algorithm, evaluated over 100 valid placements per algorithm–assembly pair. The metric is the sum, across all trajectory segments, of the largest joint rotation (degrees) in each segment. Because execution time scales linearly with joint angular speed, this measure is proportional to actual runtime.

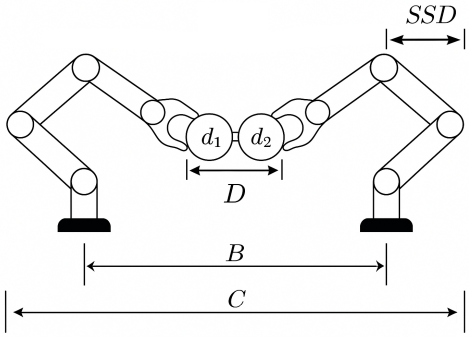

Fig. 4: Dimensions of a robotic cell. B is the distance between the bases of the robotic arms. C is the maximal width occupied by the robotic cell. SSD is the guaranteed maximal deviation of a single arm beyond its base. d_1 and d_2 are the maximal diameters of the sub-assemblies. $D = d_1 + d_2$.

As demonstrated in Section V, valid trajectory placements using two robotic arms occur much more frequently than when using a single arm. Therefore, we can limit our analysis to trajectory placements on the plane that is normal to the line connecting the bases of the two arms and passes through the midpoint between them. We assume that after each step, the entire assembly is translated back to this plane (in practice, we combine this back-shift with the step itself). This allows the analysis of the robotic cell width to depend solely on the lengths of the robotic arms (arm_length), the distance between the bases of the arms (B), and the combined diameters of the assembly pieces (D) (see Fig. 4). The length D can be calculated simply as the sum $d_1 + d_2$ of the diameters of the two sub-assemblies, though a tighter bound may assess the diameter of the assembly as a whole at each step. Let C represent the width of the robotic cell. Then:

$$C = \frac{B}{2} + \frac{D}{2} + arm\_length. \tag{8}$$

We assume that the distance between the bases of the robotic arms is larger than the assembly diameter, $B \geqslant D$, otherwise further restrictions on the trajectory may be required. Hence C, the maximal width occupied by the robotic cell, as a function of B, obtains its minimum when $B = D$:

$$C_{min} = D + arm\_length. \tag{9}$$

Further, in case the placements of the arms in the robotic cell are pre-defined and we are only interested in the single-side deviation (SSD) of the robotic arms out of the given robotic cell, we receive:

$$SSD = \frac{D}{4} - \frac{B}{4} + \frac{arm\_length}{2}. \tag{10}$$

V. Experiments and Results

In this section we demonstrate the advantage of using *CPW-IK* and *GPW-IK* with two robotic arms over a single arm. A video compiling executions of all assemblies is available on our project page.

A. Dataset

We used the tight assembly dataset presented in [10], and compared the results of using two arms to those using a single arm. The four assemblies in that dataset were selected from a larger dataset [24], keeping one challenging sample from each family. These are tight assemblies that require non-trivial combination of translation and rotation in order to be assembled, hence proved to be the hardest for the algorithms tested. In this work, as a proof of concept, we demonstrate the solutions, and measure the results, of four representative samples.

B. Implementation Details

We used the TR-RRT [25] motion planning algorithm for finding a valid trajectory for the free-flying assemblies.

We demonstrate our methods on a pair of *Universal Robots UR5e* robotic arms, equipped with *Robotiq 2F-85* grippers.

The rest of the process is as follows. We begin with a free-flying trajectory $\bar{\gamma}$ and continue with running CPW-IK for a single arm, CPW-IK for two arms, or GPW-IK (for two arms) seeking valid placements. We continued with valid placements trials until 100 valid placements were found for each combination of an assembly and an algorithm. We then design the parts in *Blender*, add a small mark or cutout at the desired grasping point, which is defined manually, and print them using *Creality Ender-3 S1* $3D$ printer with 0.2 mm resolution. Finally, we grasp the parts using the end effectors and execute the valid trajectories in the real world as can be seen in Fig. 1 and Fig. 2.

TABLE I: Path quality. The average Hausdorff distance over 100 valid placements, presented as total radians traveled per time step, for each assembly. *Error reduction*: percentage decrease in average Hausdorff distance when using two arms compared to a single arm. The *Average* column reports an average only for the improvement row (percentage gains); averages of the raw per-assembly values are not shown because they are not directly meaningful.

Assembly	az	abc	16505	06397	*Average*
Single-arm	0.035	0.041	0.028	0.121	—
Two-arms	0.027	0.023	0.017	0.103	—
Error reduction	**22.9%**	**43.9%**	**39.3%**	**14.9%**	**30.2%**

C. Results

The experiments demonstrate three main findings. First, the total assembly time (*makespan*) is consistently lower when using two robotic arms than with a single arm. Fig. 3 presents the complete set of results, including the improvement in the best solution found, the distribution of valid solutions, and the median performance. Relative to the *average* solutions, the makespan is reduced by 15.9% for CPW-IK and by 50.2% for GPW-IK, while the improvement over the *best* solutions is 14.4% for CPW-IK and 39.5% for GPW-IK. Note that these best-solution improvements do not reflect the full potential of GPW-IK, as discussed below and in Table II, because on average it explored less than one third of the placements examined by the single-arm baseline.

Second, the quality of the paths, in terms of one-sided Hausdorff distance, is improved, as can be seen in Table I. This means that the actual trajectory traveled by the assembly pieces when manipulated using two robotic arms, is closer to the desired free-flying trajectory given, than in the case of a single arm.

TABLE II: The table presents successful placements rate. *Improvement* is the ratio between the two-arm success rate algorithms and the single-arm one. The *Average* column reports an average only for the improvement rows (multiplicative gains); averages of the raw per-assembly values are not shown because they are not directly meaningful.

Assembly	az	abc	16505	06397	*Average*
Single-arm	0.021	0.124	0.194	0.033	—
CPW-IK	0.225	0.253	0.286	0.235	—
Improvement	**x10.71**	**x2.04**	**x1.47**	**x7.12**	**x5.34**
GPW-IK	0.156	0.128	0.245	0.150	—
Improvement	**x7.43**	**x1.03**	**x1.26**	**x4.55**	**x3.57**

Third, the trajectory placement algorithm performs much faster for CPW-IK and GPW-IK relative to the single-arm algorithm, as demonstrated in Table II in terms of frequency of valid trajectories. That was for finding 100 valid placements. This implies that if we provide equal running times to all algorithms (rather than wait for 100 valid placements), the improvement in assembly time is expected to be even larger (as the two-arm algorithms will have time to test a larger amount of placements).

VI. Discussion

Our experiments confirm that coordinated dual-arm planning dramatically reduces assembly makespan and improves trajectory accuracy compared to a single-arm baseline.

This work introduced GPW-IK, a new dual-arm planning algorithm with a provable $>50\%$ reduction in average assembly time and demonstrated its effectiveness on physical hardware.

These results highlight the practical relevance of our framework for Industry 4.0 manufacturing cells and motivate its adoption in time-critical assembly lines.

We continue with this research in two directions. First, is to further accelerate the simultaneous work of two arms by leveraging global information rather than only that of the current segment. Second, there are assembly operations involving three objects, which require coordinated manipulation of two arms in order to complete the assembly (see Fig. 5). We plan to use the algorithm and software package developed for this project to solve such three-handed assembly problems.

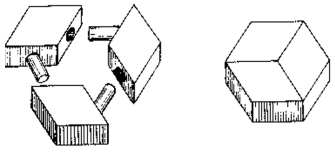

Fig. 5: Conceptual illustration of a three-handed assembly (adapted from johnrausch.com); future work will extend our method to such cases.

References

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

[2] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1, pp. 335–349, 2012.

[3] J. Fan and J. Dong, "Intelligent virtual assembly planning with integrated assembly model," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, vol. 5, 2003, pp. 4803–4808 vol.5.

[4] K.-C. Ying, P. Pourhejazy, C.-Y. Cheng, and C.-H. Wang, "Cyber-physical assembly system-based optimization for robotic assembly sequence planning," *Journal of Manufacturing Systems*, vol. 58, pp. 452–466, 2021.

[5] J. Michniewicz, G. Reinhart, and S. Boschert, "CAD-based automated assembly planning for variable products in modular production systems," *Procedia CIRP*, vol. 44, pp. 44–49, 2016.

[6] 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.

[7] M. Soori, R. Dastres, B. Arezoo, and F. K. G. Jough, "Intelligent robotic systems in industry 4.0: A review," *Journal of Advanced Manufacturing Science and Technology*, pp. 2 024 007–0, 2024.

[8] T. Lohi, S. Soutukorva, and T. Heikkilä, "Programming of skill-based robots," in *2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2024, pp. 1–7.

[9] M. Daneshmand, F. Noroozi, C. A. Corneanu, F. Mafakheri, and P. Fiorini, "Industry 4.0 and prospects of circular economy: A survey of robotic assembly and disassembly," *CoRR*, vol. abs/2106.07270, 2021.

[10] D. Livnat, Y. Lavi, and D. Halperin, "A full-cycle assembly operation: From digital planning to trajectory execution using a robotic arm," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 9184–9191.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[13] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, vol. 3. IEEE, 2003, pp. 4420–4426.

[14] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[15] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.

[16] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT*-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[17] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

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

[19] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation," *IEEE Robotics Autom. Lett.*, vol. 4, no. 2, pp. 277–283, 2019.

[20] O. Salzman, M. Hemmer, and D. Halperin, "On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages," *IEEE Trans Autom. Sci. Eng.*, vol. 12, no. 2, pp. 529–538, 2015.

[21] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3965–3972.

[22] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021.

[23] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[24] 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.

[25] D. Livnat, M. M. Bilevich, and D. Halperin, "Tight motion planning by riemannian optimization for sliding and rolling with finite number of contact points," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 333–14 340.

[26] S. Kucuk and Z. Bingul, "Inverse kinematics solutions for industrial robot manipulators with offset wrists," *Applied Mathematical Modelling*, vol. 38, no. 7-8, pp. 1983–1999, 2014.

[27] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 4, pp. 313–336, 1994.

[28] J. Villalobos, I. Y. Sanchez, and F. Martell, "Singularity analysis and complete methods to compute the inverse kinematics for a 6-dof ur/tm-type robot," *Robotics*, vol. 11, no. 6, 2022.

[29] S. I. Y. Villalobos Jessica and M. Fernando, "Alternative inverse kinematic solution of the ur5 robotic arm," in *Proceedings of the Latin American Congress on Automation and Robotics*. Springer, 2021, pp. 200–207.

[30] L.-T. Schreiber and C. Gosselin, "Determination of the Inverse Kinematics Branches of Solution Based on Joint Coordinates for Universal Robots-Like Serial Robot Architecture," *Journal of Mechanisms and Robotics*, vol. 14, no. 3, p. 034501, 11 2021.

[31] T. Ho, C.-G. Kang, and S. Lee, "Efficient closed-form solution of inverse kinematics for a specific six-dof arm," *International Journal of Control, Automation and Systems*, vol. 10, pp. 567–573, 2012.

[32] J. Li, H. Yu, N. Shen, Z. Zhong, Y. Lu, and J. Fan, "A novel inverse kinematics method for 6-dof robots with non-spherical wrist," *Mechanism and Machine Theory*, vol. 157, p. 104180, 2021.

[33] M. H. FarzanehKaloorazi and I. A. Bonev, "Singularities of the typical collaborative robot arm," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 51814. American Society of Mechanical Engineers, 2018, p. V05BT07A086.

[34] D. Rakita, B. Mutlu, and M. Gleicher, "Stampede: A discrete-optimization method for solving pathwise-inverse kinematics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3507–3513.

[35] Y. Wang, C. Sifferman, and M. Gleicher, "Iklink: End-effector trajectory tracking with minimal reconfigurations," 2024. [Online]. Available: https://arxiv.org/abs/2402.16154

[36] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *2018 second IEEE international conference on robotic computing (IRC)*. IEEE, 2018, pp. 137–142.

[37] W. Wan, K. Harada, and K. Nagata, "Assembly sequence planning for motion planning," *Assembly Automation*, vol. 38, no. 2, pp. 195–206, 2018.

[38] 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," *arXiv preprint arXiv:2407.08028*, 2024.

[39] T. Cohn, S. Shaw, M. Simchowitz, and R. Tedrake, "Constrained bimanual planning with analytic inverse kinematics," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6935–6942.

[40] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, "Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 888–901, 2021.

[41] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga, "C-space tunnel discovery for puzzle path planning," *ACM Trans. Graph.*, vol. 39, no. 4, aug 2020.