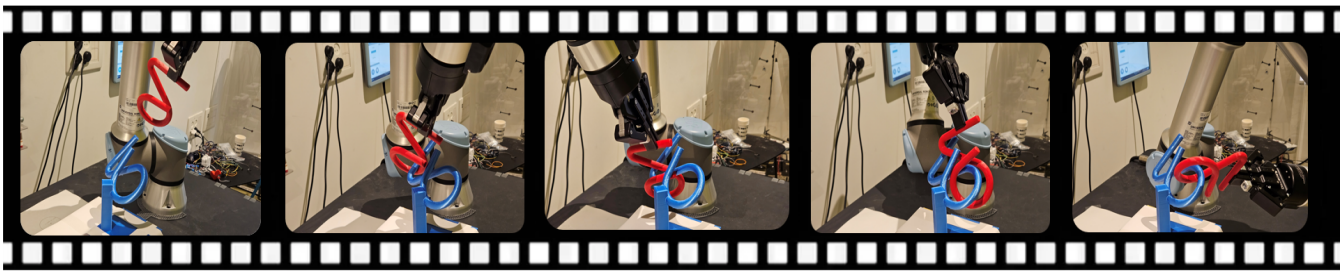
Full-Cycle Assembly Operation: From Digital Planning to Trajectory Execution by a Robot Arm

Dror Livnat*† Yuval Lavi*† Dan Halperin†



***Abstract*— We present an end-to-end framework for planning tight assembly operations, where the input is a set of digital models, and the output is a full execution plan for a physical robotic arm, including the trajectory placement and the grasping. The framework builds on our earlier results on tight assembly planning for free-flying objects and includes the following novel components: (i) post processing of the free-flying paths to relax the tightness (where possible) and smooth the path, (ii) employing analytic path-wise inverse kinematic (IK) solutions with IK-branch switching where needed, (iii) trajectory placement search based on the above path-wise IK, to determine feasible arm paths, and (iv) coping with the grasping challenge. The framework provides guarantees as to the quality of the outcome trajectory. For each component we provide the algorithmic details and a full open-source software package for reproducing the process. Lastly, we demonstrate the framework with tight and challenging assembly problems (as well as puzzles, which are planned to be hard to assemble), using a UR5e robotic arm in the real world and in simulation. See the figure at the top for a physical UR5e assembling the alpha-z puzzle (known to be considerably more complicated to assemble than the celebrated alpha puzzle). Full video clips of all the assembly demonstrations together with our open source software are available at `https://github.com/TAU-CGL/Full-Cycle-Assembly-Operation`**

I. Introduction

Assembly planning, as well as other robotic manipulation tasks such as painting or welding are a central pillar in the fourth industrial revolution [1]–[3]. Assembly is a fundamental process in manufacturing, where separate parts are put together to create a final product. *Assembly Planning* is the process of determining a sequence of motions for this assembly. It also offers valuable feedback to CAD designers [4]. In manufacturing industry, assembly consumes about half of the production time and 20% of total manufacturing costs [5], [6]. The growing demand for mass customization requires flexibility in production, often achieved during product assembly [7]. Consequently, Assembly Planning and robotic assembly have become prominent fields of research [8], [9].

With millions of industrial robots operating in factories, and them being one of the most quickly growing industrial sectors [3], the challenge of translating a digital plan of the parts into a final robotic arm trajectory becomes a critical task. Yet, the control and programming of the commercially available industrial robots are limiting factors for their effective implementation, especially for dynamic production environments or when complex applications are required [10]. There are motion-planning solutions in the Cartesian space for the sub-assemblies as free-lying objects, such as [11]–[13], and our previous work [14]. However, it is crucial to provide a solution in the robot joint-space, as inverse kinematics (IK) for configurations along the free-flying[1] path do not guarantee a continuous trajectory in the joint space due to singularities, where two neighboring configurations in the Cartesian space might require two far apart C-space configurations, and different IK branches (see Section III-C for details).

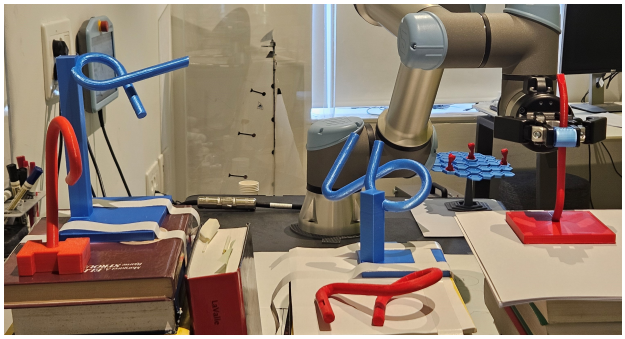


Fig. 1: Three tight assemblies physically assembled using our framework. On the left, the famous alpha puzzle. In the middle, the much harder alpha-z puzzle. On the right, assembly no. 16505 from [12] in its assembled state, with which we demonstrate trajectory search with constraints. The video clips for all assemblies are available at `https://github.com/TAU-CGL/Full-Cycle-Assembly-Operation`

We are not aware of any existing end-to-end software

*Equal first contribution.

†Blavatnik School of Computer Science, Tel-Aviv University, Israel. This work has been supported in part by the Israel Science Foundation (grant no 2261/23), by NSF/US-Israel-BSF (grant no. 2019754), by the Blavatnik Computer Science Research Fund, and by the Shlomo Shmelzer Institute for Smart Transportation at Tel Aviv University.

[1]By *free-flying* we refer to objects that are neither fixed by a fixture nor attached to a robotic arm.

Manuscript 1526 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

solution that receives a digital design of sub-assemblies on one end, and outputs a joint-based trajectory and placement to solve the assembly—which actually work in the real world—on the other. Some of the building blocks are in place. In this work we address and add the missing pieces, and put all the ingredients together.

We break this challenge into a few steps, starting with the free-flying objects trajectory planning. Sampling based approaches dominate the field for more than two decades, with Probabilistic Roadmaps (PRM) [15] and Rapidly-Exploring Random Trees (RRT) [11] guaranteeing *probabilistic completeness*.[2] [16], [17] followed by numerous variants aimed at improving parameters such as computational speed [18]–[20], path quality [19], [21], or guaranteeing anytime performance [20], [22]. Various strategies were used to attack the remaining challenge of *tight*[3] assembly planning, including applying a bridge-test for samples [23], [24], sampling sub-manifolds of the C-space rather than just points [25], [26], learning-based approaches [27], [28], approaches based on the specific geometric features of the elements [13], [29], compliant motion planning [30]–[34], physical-simulation based methods [12], [35], and learning [14], [36]–[41].

Most of the above approaches were tested in simulation, and they allow for some minimal penetration between the objects. Having real world implementation in mind, these penetrations should be avoided. We use a variant of the *retract* function from our previous work [14] to eliminate those penetrations. We then smooth this path by removing configuration points and replacing them with newly interpolated ones, as long as the newly created configurations do not create new penetrations.

Then comes the *Inverse Kinematics* (IK) phase, transferring these workspace configurations to the joint-space. There are two main approaches to IK. One is numeric [42], [43], which may suffer from instability around robotic arm singularities [44] as well as possible inconsistencies of the returned branch. The other one is analytic [45], [46], available for the UR-like arms, providing analysis and control over the returned branch [44], and is faster and more accurate, especially near singularities [47], [48]. In this work we focus on the Universal Robots geometry, which is very common among today's cobots, and is one of the fastest growing in general [58] since its launch in 2008. However, isolated IK solutions do not guarantee continuous end-effector trajectory due to robot singularities, as well as internal and external obstacles. Additional work was done on path-wise IK [49]–[51], although results are forced to a single robot placement, algorithm allows variance around the desired configurations, and numeric IK is in use, which makes solutions more challenging around singularities.

Last is the challenge of *trajectory placement*. This is similar to the well studied *robot placement* problem [52], where valid locations of the robot's base are searched for each desired pose, or a set of poses, of the robot's end effector. Further work was done on mapping and testing robot placement for multiple isolated configurations [52] in order for the robot to be able to complete a path. However, our challenge is to place a full continuous trajectory, which may be considered as placing a $6D$ curve in a complex $6D$ environment.

Works we have found that tackle the entire framework differ from our own in their settings and the type of assemblies they handle. One such work solves only the translational ($3D$) case [53]. A more recent one [54] provides a dataset of peg-in-a-hole type of problems, where the pegs are in vertical orientation, taking an RL approach, using a camera, to cope with them. Our work focuses on providing a full framework, from digital plans of sub-assemblies to real world execution, of tight assembly problems including non-trivial combination of translation and rotation.

The contribution in this work is as follows: (1) a framework that takes in digital designs of sub-assemblies, and outputs an execution plan for their assembly in the physical world using a robotic arm, which is the first framework and implementation that covers the whole range from planning to actual execution to the best of the authors' knowledge, (2) guarantees, in terms of Hausdorff distance, bounding the error between such executed trajectory and the desired one, (3) tailored relaxation and smoothing functions appropriate for bridging the gap between the free-flying digital objects trajectory to the robotic arm physical one, (4) closed form IK for a continuous path deploying dynamic programming to maintain or switch branches, coping with singularities, (5) placement of the $6D$ continuous workspace trajectory relative to the robotic arm, which is equivalent to placement of a $6D$ curve in a complex $6D$ environment, (6) demonstration of the framework with a physical robot executing tight assembly tasks, and (7) Implementation of all the above in the form of an open source code package.

II. Problem Statement

In this work we tackle the problem of receiving the design of two rigid bodies[4] to be assembled, together with desired start and goal relative positions. The output is detailed instructions of how a robotic arm may perform the assembly task, including trajectory placement, grasping poses, and trajectory of the robotic arm in term of joint angles along the process.

More formally, we start with two rigid bodies $B_1, B_2 \subseteq \mathbb{R}^3$, which we wish to assemble. We assume that B_1 is fixed at some pose $\bar{q}_{\text{static}} \in SE(3)$, and that B_2 is the object we wish to move from configuration $\bar{q}_{\text{start}}$ to configuration $\bar{q}_{\text{goal}}$ in $SE(3)$. From here on, we refer to B_1 as the *static body* or *static sub-assembly*, and to B_2 as the *dynamic body* or the *dynamic sub-assembly*.

The configuration space, C-space for short, of the robotic arm (a 6 joints *UR5e* by Universal Robots), $C \subseteq \mathbb{R}^6$ such that $c \in \mathcal{C}$ is the robotic arm configuration in joint-space. That is $c = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$, $0 <= \theta_j < 4\pi$ is a set of joint angles of the arm. The forward kinematics function f_g maps a C-space configuration $c \in C$ to a workspace pose $\bar{q} = (f_g(c))$ given some linear transformation g representing the grasping pose.

[2]Probabilistic completeness is the property that as more samples are added, the probability that the planner fails to find a path, if one exists, asymptotically approaches zero.

[3]By *tight* we refer to motion planning instances where any collision-free path from start to target must go through a point with very low clearance, making it hard to solve by standard planning techniques.

[4]It is convenient for the purpose of description and analysis to think of the B_i's as open sets in $\mathbb{R}^3$, as in particular we wish to let them touch each other during the (dis)assembly operation.

Let ℓ_j^i be the j'th link of the robot when in configuration c_i. We say that $c_i \in \mathcal{C}_{\text{forbid}} \subseteq \mathcal{C}$ if any two of the robot links ℓ_j^i and the two rigid bodies are intersecting.

We define the *free region* as all configurations that yield no intersection of the robot and the objects,

$$\mathcal{C}_{\text{free}} = (\mathcal{C} \setminus \mathcal{C}_{\text{forbid}}). \tag{1}$$

Our problem is then finding a grasp g together with a collision free path from $c_{\text{start}} \in \mathcal{C}_{\text{free}}$ to $c_{\text{goal}} \in \mathcal{C}_{\text{free}}$, i.e., to find some path $\gamma : [0,1] \to \mathcal{C}_{\text{free}}$ such that $\gamma(0) = c_{\text{start}}$ and $\gamma(1) = c_{\text{goal}}$, such that $\bar{q}_{\text{start}} = f_g(c_{\text{start}})$ and $\bar{q}_{\text{goal}} = f_g(c_{\text{goal}})$.

In this work we restrict ourselves to paths γ that are piecewise linear, that is, a sequence:

$$c_{\text{start}} = c_0, c_1, \ldots, c_{N-1}, c_N = c_{\text{goal}}, \tag{2}$$

and the segment between each pair of consecutive configurations in the sequence, such that γ is contained in $\mathcal{C}_{\text{free}}$.

III. A FRAMEWORK FOR TIGHT ASSEMBLY OPERATIONS

This section describes the main steps of the complete pipeline from the digital plan of the sub-assemblies to the actual execution by a physical robotic arm.

A. Motion planning for free-flying objects

We start with a free flying tight assembly problem, presented as digital modeling of two free-flying bodies B_1 and B_2, as well as start and goal positions $\bar{q}_{\text{start}}$ and $\bar{q}_{\text{goal}}$ as presented in Section II. We deploy an appropriate motion planning algorithm, e.g., as described in [12]–[14]. Specifically, in this work we the TR-RRT algorithm from our previous work [14], as it best performed on the dataset presented in [12], from which we extracted the demonstrations in the current work. The output of this phase is a piecewise linear path $\bar{\gamma}$, that is, a collision free path 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

$$\bar{q}_{\text{start}} = \bar{q}_0, \bar{q}_1, \ldots, \bar{q}_{N-1}, \bar{q}_N = \bar{q}_{\text{goal}}, \tag{3}$$

and the segment between each pair of consecutive configurations in the sequence, such that $\bar{\gamma}$ is contained in $\bar{\mathcal{C}}_{\text{free}}$.

B. Relaxation and smoothing

The output path $\bar{\gamma}$ from the previous section has two properties that are undesirable for the next phases.

First, free-flying motion planning algorithms usually allow some small penetration between the sub-assemblies. This works perfectly in simulation, but is challenging for the physical world. Hence, we start by leveraging a variant of the *retract* function from [14], which is based on the gradient of the SDF (the $6D$ direction that maximizes the separation of the sub-assemblies), and slightly shift and rotate every configuration $\bar{q}_i$ to $\bar{q}_i'$, such that the distance between the sub-assemblies is either positive (for more than 95% of the cases, which are not in the narrow passages of the path), or the penetration is smaller than $10^{-5}m$, which is smaller than the resolution of our manufacturing means (3D printer). This relaxation also makes the joint-space trajectory γ less sensitive to calibration or manufacturing errors.

The second undesirable property of the free-flying path $\bar{\gamma}$ is that it may be, as is often the case with sampling-based plans, unnecessarily long and jagged. For this we carry out a post processing step of shortening and smoothing the trajectory $\bar{\gamma}$ in the workspace, in order to remove redundant parts and speed up the execution. In this part we remove every other configuration along $\bar{\gamma}$ as long as the interpolated configuration between the two remaining end configurations does not generate a new penetration. We repeat this process a few times. However, in order to avoid later penetrations due to interpolation in the joint-space trajectory γ over too long edges, we actually replace every removed configuration in the workspace trajectory $\bar{\gamma}$ with a new configuration, which is the interpolation of its two neighbors. This way we can create workspace edges as short as we wish, hence limiting the potential for penetration in the interpolation to be performed in the joint space. For sake of simplicity we reuse the symbol $\bar{\gamma}$ to represent the path after relaxation and smoothing.

C. CPW-IK: Continuous path-wise IK, and branch switching

This step addresses one of the major and recurring difficulties in transfer of the trajectory from free-flying sub-assemblies to sub-assemblies manipulated by a robot arm: The behaviour of the trajectory in the C-space, namely in the joint-angles space. Quite often, a valid trajectory in the workspace has no feasible counterpart in joint space due to a variety of reasons including the lack of relevant inverse kinematic solutions, singular configurations, or self collision of the robot links. Moreover, the motion between configurations that are close-by in the workspace may translate to convoluted and long motions in joint space. We wish to generate a path in C-space that will make the end-effector track the path $\bar{\gamma}$ up to some pre-defined error ϵ. The resulting joint-space trajectory should be continuous and collision-free. We denote the sub-assembly manipulated by the robotic arm by A. In order to define the error of a generated path we first look at $\partial A \subseteq \mathbb{R}^3$, the boundary of A. For a generated path γ, each point $p \in \partial A$ follows some trajectory $\gamma_p \subseteq \mathbb{R}^3$. Similarly, for the original path $\bar{\gamma}$, p follows a trajectory $\bar{\gamma}_p \subseteq \mathbb{R}^3$. We define the error of p under γ to be the one-sided Hausdorff distance [55] from γ_p to $\bar{\gamma}_p$

$$d(\gamma_p, \bar{\gamma}_p) = \max\{\min\{d(x,y) : y \in \bar{\gamma}_p\} : x \in \gamma_p\}.$$

Informally, while following γ, this is the furthest that p gets from its designated trajectory $\bar{\gamma}_p$. Now, we define the error of a generated trajectory to be the furthest that any $p \in \partial A$ gets from its designated trajectory:

$$\text{Err}(\gamma) = \max\{d(\gamma_p, \bar{\gamma}_p) : p \in \partial A\}.$$

Recall that $\bar{\gamma}$ is piece-wise linear in the Cartesian space, and γ is piece-wise linear in the robot's joint space. We call the vertices of γ *anchor configurations* and denote them by c_i. We wish to bound $\text{Err}(\gamma)$ by a linear function of the largest L^1 distance between adjacent anchor configurations of γ.

$\text{Err}(\gamma)$ is bounded by the maximal total distance any point $p \in \partial A$ travels during the linear transition of a robot C-space configuration c from c_i to c_{i+1}. This is true since for the anchor configurations the generated and original configurations are equal. For a given distance δ between two anchor configurations, the scenario in which a point $p \in \partial A$ undergoes the largest distance is when the arm is fully extended, parallel to the base, and the base joint moves by δ. We denote the maximal distance of a point in our moving system (the robot and the dynamic body attached to it) from the robot's base by D. Lastly, for a trajectory γ with a maximal L^1 distance of δ between adjacent anchor configurations:

Manuscript 1526 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

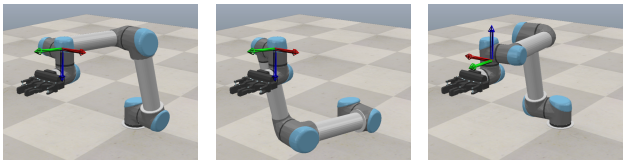
$$\mathrm{Err}(\gamma) \leq \max\{\int_{t_{c_i}}^{t_{c_{i+1}}} \gamma_p(t)dt : p \in \partial A, i \in [N]\} \leq \delta \cdot D, \quad (4)$$

where t_{c_i} denotes the time that γ reaches c_i, and we get the required linear bound with a constant D. Using this bound, we can control $\mathrm{Err}(\gamma)$ by changing δ.

As for self-collisions, the bound in Inequality (4) holds also for every point on the boundary of the moving system, including the robot's links. For every pair of adjacent anchor configurations, c_i, c_{i+1}, a point p on the boundary of the arm travels a distance of at most $\delta \cdot D$. Hence, throughout this motion, p keeps a distance of at most $\frac{\delta \cdot D}{2}$ from at least one of $\gamma_p(t_{c_i}), \gamma_p(t_{c_{i+1}})$. Therefore, by ensuring that each anchor configuration is collision-free with $\delta \cdot D$ clearance, we guarantee that the entire trajectory is collision-free.

To achieve that, our first building block is a single pose analytic IK solver. In this work we use the IK solution suggested by [45].

We start with a short reminder regarding inverse kinematics *branches* (see [46] for details). For each end-effector pose, the $6-DoF$ UR-like robotic arm has up to 8 possible different IK solutions. These solutions differ from each other by three parameters: shoulder left/right, elbow up/down and wrist left/right-hand rule, as can be seen in Figure 2.

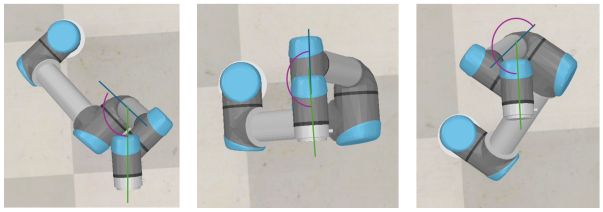


(a) (b) (c)

Fig. 2: Different IK branch types of a *UR5e*. A right-handed coordinate frame is attached to each wrist pose. The wrist is considered to be in a right-hand configuration if the wrist 2 rotation axis can be found using the right-hand rule on axes wrist 1 and wrist 3. In (a) and (b) the wrist's axes align with the right-handed coordinate frame, thus they are in a right-handed wrist branch. In (c) the middle wrist axis, called wrist 2, is pointing in the opposite direction hence the pose is of a left-handed wrist branch. The other parameters are: (a)/(c) show a left/right shoulder configuration, respectively; (a)/(b) show an up/down elbow configuration.

Every combination of these three parameters defines an IK branch. We denote these parameters by $i,j,k \in \{-1,1\}$, and specify their values when carring out an analytic IK calculation. The shoulder's side is determined by i, the wrist by j, and the elbow by $i \cdot k$. In [46], they also show that given a joint-space configuration of the arm, these parameters can be easily extracted.

Having introduced the concept of IK branches, a naive approach to computing a trajectory in joint space would be to choose a specific branch and apply IK for each pose along the path using fixed i,j,k parameters corresponding to that branch. Unfortunately, some trajectories are physically infeasible using a single branch as can be seen in Figure 3 and in a full video in the supplementary material.



(a) (b) (c)

Fig. 3: Three key frames from a straight line trajectory performed by a *UR5e* arm. No single branch is feasible in both the left and right frames. Using the wrist singularity (b), the arm can switch from the right-hand wrist configuration (a) to the left-hand wrist configuration (c) and successfully complete the motion. The full video is provided in the supplementary material.

To overcome this issue, we introduce *branch switching* in our solution. For each anchor point in the given trajectory, we calculate the IK solution under every feasible branch. We build a layered Directed Acyclic Graph $G=(V,E)$, where layer L_k contains the feasible IK solutions of pose k in the trajectory as nodes. We connect node $a \in L_k$ with a directed edge to node $b \in L_{k+1}$ if and only if their L^1 distance in C-space is smaller than some predefined threshold δ. We let the weight $w(e)$ of each edge $e \in E$ be this distance. Next, we wish to find a path from the first pose to the last one, while guaranteeing the best bound we can on $\mathrm{Err}(\gamma)$. This is done by adding two dummy nodes s,t to the graph, and connecting them with zero-weight edges to the nodes of L_1 and L_N respectively, where N is the number of anchor points in the trajectory. We the perform a search for a directed path in G from s to t with the minimum maximally-weighted edge. Let $\Pi_G(s,t)$ denote all the paths from s to t in G, then:

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

We find γ by replacing the sum in Dijkstra's shortest path algorithm [56] with the max function. For a trajectory $\gamma \subseteq \mathcal{C}$, we denote the largest distance between two neighboring poses in the trajectory by δ_γ. Using this variant of Dijkstra's algorithm we get γ with the optimal δ_γ value. Thus, we get the best error guarantee from Inequality (4). If $\delta_\gamma \leq \frac{\epsilon}{D}$, we get the desired $\mathrm{Err}(\gamma) \leq \epsilon$. We note that a similar graph and dynamic-programming approach appear in [49] but with a few major differences: A layer in their graph is a collection of nearby numeric IK solutions obtained by adding noise to each intermediate configuration in the path, with no reference to branches, and, the context is of a constant placement. In contrast, our graph uses closed-form exact IK configurations including the 8 IK branches (if available), enabling for the quick trajectory placement.

A demonstration of a simplified version of the graph that we construct for the trajectory from Figure 3 can be seen in Figure 4. A pseudo-code of our CPW-IK algorithm is presented in Algorithm 1.

D. Trajectory placement

The outcome of the first two steps of our framework (Sections III-A and III-B) is a trajectory $\bar{\gamma}$ for the dynamic

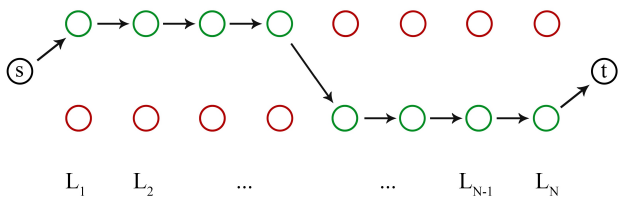


Fig. 4: A simplified version of the IK layered graph of the trajectory from Figure 3. The upper nodes refer to the right-hand wrist solutions, and the lower to the left-hand wrist configurations. The other branches were omitted for clarity.

Algorithm 1 CPW-IK

Input: end-effector trajectory $\bar{\gamma}$, distance threshold $\delta > 0$
Output: joint-space trajectory ρ
$L_1,...,L_N \leftarrow [\ ]$
$V \leftarrow [s,L_1,L_2,...,L_N,t]$
$E \leftarrow [\ ]$
$G \leftarrow (V,E)$
for $p \in \bar{\gamma}$ **do**
  for $i,j,k \in \{-1,1\}^3$ **do**
    $v_{p_{ijk}} \leftarrow IK(p,i,j,k)$
    if $v_{p_{ijk}}$ is not $null$ and *collision_free*$(v_{p_{ijk}})$ **then**
      add $v_{p_{ijk}}$ to L_p
      for $v' \in L_{p-1}$ **do**
        $\Delta \leftarrow \|v_{p_{ijk}} - v'\|$
        if $\Delta < \delta$ **then**
          add $(v', v_{p_{ijk}}, \Delta)$ to E
add 0-weight edges from s to L_1
add 0-weight edges from L_N to t
$\rho \leftarrow$ shortest_path(G,s,t)
return ρ

sub-assembly, which solves our assembly operation in the free-flying object setting. Notice that $\bar{\gamma}$ is a curve in six-dimensional space.

The objective here is to find a placement (position and orientation) for the Cartesian trajectory $\bar{\gamma}$, that will enable a successful solution of the continuous path-wise IK (CPW-IK, Section III-C). As mentioned in the Introduction, this problem is similar to the well studied *robot placement* problem [52], however, here the challenge is to place a full continuous trajectory.

To place the trajectory, we begin by randomly sampling a $6D$ starting pose $\bar{q}_{\text{start}}$ for the manipulated sub-assembly, and derive the starting pose of the static sub-assembly from it, as the initial relation between them is known. Once we have an initial pose candidate $\bar{q}_{\text{start}}$, we translate and rotate the $6D$ curve $\bar{\gamma}$ into $\bar{\gamma}_{\bar{q}_{\text{start}}}$, where the initial pose matches the sampled one. We then run the CPW-IK algorithm on $\bar{\gamma}_{\bar{q}_{\text{start}}}$. If successful, a valid placement for the trajectory was found, and we are done. Otherwise, we sample a new initial pose and repeat the process.

As environment or application may pose various constraints on the assembly, such as available space in the cell, or preferred orientation of the objects. Therefore, the trajectory-placement function supports bounding each of the 6 dimensions of the search to a a small interval or even a single point. This functionality can significantly speed up the search, when such constraints are known to the user. A demonstration of this functionality appears in constraining industrial assembly 16505 to a single orientation as can be seen in Figure 1.

To further narrow down our search, we use known kinematic limitations of the robotic arm at hand. For instance, a pose where the projection of the base of the final link on the xy plane is closer than d_4 to the robot's base is physically unreachable [57]. Additionally, the distance between the base of the final link and the base of the robot can not exceed $a_2+a_3+\sqrt{d_4^2+d_5^2}$ (a_i,d_i are DH parameters as illustrated in [44]).

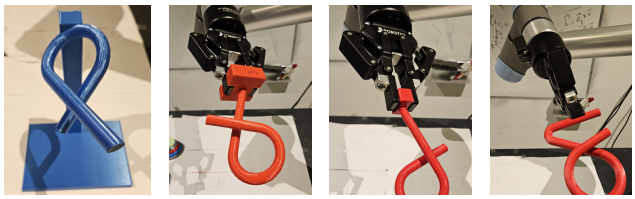
For each path $\bar{\gamma}_{\bar{q}_{start}}$ generated during the trajectory placement, we are guaranteed that our CPW-IK solver gives us the optimal feasible $\gamma_{\bar{q}_{\text{start}}}$ in terms of δ, if one exists.

Remark. *Trajectory placement* is similar to what is referred to in the literature as *robot placement*, albeit with two main differences. Robot placement looks for a suitable placement for the robot from which it can access a given configuration $\bar{q} \in SE(3)$, or a small set of such configurations. In trajectory placement the robot placement is given, and we seek a feasible pose for the entire trajectory—an infinite set of configuration points—according to the given robot placement. The second difference is that keeping the robot still, and moving the trajectory around, provides the flexibility of coping with more complex scenarios such as additional constraints on the trajectory pose (see, e.g. the solution for assembly no. 16505 in Figure 1, where orientation was decided based on ease of printing, and given to the algorithm as input), and further to place more than a single trajectory relative to the same robotic arm, which is typical in manufacturing environments.

E. Grasping

The result of the *trajectory placement* phase is configurations $\bar{q}_{start}$ for the dynamic sub-assembly and $\bar{q}_{static}$ for the static one. When shifting to the real world, a physical fixture (or an additional robotic arm, when available) is required in order to fix the static body in its pose. We place it in simulation in $\bar{q}_{\text{static}}$, then adding a vertical fixture at a point where the dynamic object does not cross throughout the assembly, usually the end farthest from the arm base. We create a base aligned with the robot base frame as can be seen in Figure 5a, so locating it in the real world is easy. For the dynamic sub-assembly, the chosen grasping point and orientation define a relative grasping pose G between the frame of the end effector and the frame of the dynamic sub-assembly. The grasping position and orientation is decided after the trajectory placement phase, to ensure it is free from contact along the execution. This pose should be carefully calibrated at the beginning of the execution, and firmly maintained throughout the execution.

At first, we used to attach a large piece, complementing the robot end-effector to a box, which we refer to as *large negative*. That negative, attached to the sub-assembly at manufacturing (in fact, $3D$ printing) time, ensures precise grasping and the desired frame pose G between the sub-assembly and the end effector, as can be seen in Figure 5b. Later, in order to minimize intervention in the product and motion limitations, we switched to a minimal version of this calibration mechanism, as can be seen in Figure 5c. Lastly, in order to completely remove any motion limitations, a



(a) (b) (c) (d)

Fig. 5: Four levels of fixture or grasping to ensure relative pose of the sub-assembly: (a) simple fixture for the static sub-assembly, (b) large jig, (c) minimal jig, and (d) barely visible cutout.

cutout was formed, which requires minimal modification to the object itself, and no limitation at all on the free-flying object solution path, as can be seen in Figure 5d.

IV. Experiments

Our experiments aim to validate the proposed framework for tight assembly operations. This research addresses challenges in automated assembly planning that have not been previously tackled in a similarly broad fashion, to the best of our knowledge. Consequently, there are no direct benchmarks for comparison. We therefore demonstrate its capabilities on a variety of challenging cases, showcasing the framework's ability to handle assembly problems that require non-trivial combinations of translation and rotation in their solution. We do so using physical robots and assemblies, and in simulation.

A. Implementation details

For the first stage of free-flying objects motion planning we have used the open source software from our previous work [14], which is available at `https://bitbucket.org/taucgl/tr-rrt-public`.

The physical settings includes a Universal Robots *UR5e*, equipped with a *Robotiq 2F-85* gripper, which provides flexibility as to grasping points with minimal intervention with the grasped parts.

Once a trajectory γ is found, we run it in simulation, using *CoppeliaSim*, and choose grasping points for both sub-assemblies. We then locate the static part in the *Blender* $3D$ software tool and add a vertical fixture with base parallel to the robot base frame to ease the positioning in the physical world. We then choose a point on the dynamic part, which is farthest from contact, and form a cutout in it, digitally calculating the relative frame of the cutout for later adjustment of the trajectory. At this point we are ready for printing the two parts. To this end we have deployed a common *Ender-3 S1* $3D$ printer, with printing resolution of 0.2mm. We then locate the parts in the lab or in the simulation environment. Finally, we execute the trajectory γ generated by the algorithm.

B. Dataset

We demonstrate our work on assembly instances taken from a dataset first presented in [12], and parts of which were also used in [13] and [14]. From this dataset, we refer to two sub-groups *puzzle* and *others*, which 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 [12].

C. Evaluation

We demonstrate our framework on 6 assemblies chosen such that from every assemblies family the most difficult one is represented. As printing time of the objects was found to be a bottleneck, we present video clips of 3 instances in real life, and 3 instances in simulation. Technical information is presented in Table I, and the video clips for all 6 assemblies are available at `https://github.com/TAU-CGL/Full-Cycle-Assembly-Operation`.

TABLE I: For each assembly, # vertices($\bar{\gamma}$) represents the number of configurations in the free-flying path. *Placement* represents the average *Trajectory placement* time for finding a single valid placement in seconds, using Apple M1 Pro processor. *Ratio* represents the portion of valid trajectories out of all trajectories sampled in the search.

Assembly	alpha	az	abc	16505	09301	06397
# vertices($\bar{\gamma}$)	2,649	6,102	2,399	1,498	4,108	7,420
Placement	12.48	37.4	3.3	1.73	10.63	30.0
Ratio	0.048	0.033	0.34	0.54	0.154	0.07

V. Discussion

In this work we present, for the first time, a framework that takes in digital designs of sub-assemblies, and ends by executing their assembly in the physical world. We provide quality guarantees for the executed trajectory by bounding the error between this joint-space trajectory and the computed trajectory in the Cartesian space. We have developed dedicated relaxation and smoothing functions for bridging the gap between the trajectory of the free-flying digital objects and the trajectory we compute in joint space. We have devised a method to produce consistent path-wise IK solutions, and used it for placement search for the $6D$ continuous workspace trajectory, which is equivalent to placement of a $6D$ curve in a complex $6D$ environment. Lastly, we provide an open source software package enabling to reproduce all of the above.

The only part of the framework that still involves some manual intervention is the grasping. We took a rather simple approach there: Choose a portion (part) of the sub-assembly farthest from contact with the other sub-assembly throughout the process, and (i) form a cutout in it, if it is the dynamic sub-assembly, or (ii) add a fixture there, if it is the static sub-assembly. We aim to reduce the amount of manual intervention after further research.

In this work, the combination of the tasks at hand together with the quickness of the trajectory IK and placement, resulted in fast running times in spite of using a simple search. In future work, in order to cope with more challenging assemblies, we aim to devise a more advanced search, which in particular will include optimization for various criteria.

In this work we have focused on execution of a single assembly per robotic cell. In industry 4.0, and especially in dynamic production—e.g., assembly of a few sub-models of the same car model, or coffee machine, in the same line—placement of a few trajectories within the same environmental constraints may be required and leverage the presented *trajectory placement*. We postpone such challenges to future work.

Manuscript 1526 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

REFERENCES

[1] M. Javaid, A. Haleem, R. Singh, and R. Suman, "Substantial capabilities of robotics in enhancing industry 4.0 implementation. cognitive robotics, 1, 58–75," 2021.

[2] J. Arents and M. Greitans, "Smart industrial robot control trends, challenges and opportunities within manufacturing," *Applied Sciences*, vol. 12, no. 2, p. 937, 2022.

[3] A. Dzedzickis, J. Subačiūtė-Žemaitienė, E. Šutinys, U. Samukaitė-Bubnienė, and V. Bučinskas, "Advanced applications of industrial robotics: New trends and possibilities," *Applied Sciences*, vol. 12, no. 1, p. 135, 2021.

[4] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000.

[5] 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.

[6] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612521000042

[7] 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.

[8] 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.

[9] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584522000540

[10] P. Bilancia, J. Schmidt, R. Raffaeli, M. Peruzzini, and M. Pellicciari, "An overview of industrial robots control and programming approaches," *Applied Sciences*, vol. 13, no. 4, p. 2582, 2023.

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

[12] 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.

[13] 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. [Online]. Available: https://doi.org/10.1145/3386569.3392468

[14] 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.

[15] 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.

[16] L. E. Kavraki, M. N. Kolountzakis, and J. Latombe, "Analysis of probabilistic roadmaps for path planning," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, USA, April 22-28, 1996*. IEEE, 1996, pp. 3020–3025. [Online]. Available: https://doi.org/10.1109/ROBOT.1996.509171

[17] 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. [Online]. Available: https://doi.org/10.1109/LRA.2018.2888947

[18] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "A fast RRT algorithm for motion planning of autonomous road vehicles," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1033–1038.

[19] 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.

[20] 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.

[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[22] 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.

[23] 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.

[24] 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.

[25] 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. [Online]. Available: https://doi.org/10.1109/TASE.2014.2331983

[26] O. Salzman, M. Hemmer, B. Raveh, and D. Halperin, "Motion planning via manifold samples," *Algorithmica*, vol. 67, no. 4, pp. 547–565, 2013. [Online]. Available: https://doi.org/10.1007/s00453-012-9736-1

[27] W. Wang, L. Zuo, and X. Xu, "A learning-based multi-RRT approach for robot path planning in narrow passages," *Journal of Intelligent & Robotic Systems*, vol. 90, pp. 81–100, 2018.

[28] H. Ha, J. Xu, and S. Song, "Learning a Decentralized Multi-arm Motion Planner," in *Conference on Robotic Learning (CoRL)*, 2020.

[29] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian, "Efficient path planning in narrow passages for robots with ellipsoidal components," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 110–127, 2022.

[30] F. Dai, A. Wahrburg, B. Matthias, and H. Ding, "Robot assembly skills based on compliant motion," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE, 2016, pp. 1–6.

[31] J. Friedman, J. Hershberger, and J. Snoeyink, "Efficiently planning compliant motion in the plane," *SIAM Journal on Computing*, vol. 25, no. 3, pp. 562–599, 1996.

[32] J. De Schutter and H. Van Brussel, "Compliant robot motion I. A formalism for specifying compliant motion tasks," *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, 1988.

[33] M. A. Peshkin, "Programmed compliance for error corrective assembly," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 473–482, 1990.

[34] T. Lefebvre, J. Xiao, H. Bruyninckx, and G. De Gersem, "Active compliant motion: a survey," *Advanced Robotics*, vol. 19, no. 5, pp. 479–499, 2005.

[35] S. Zickler and M. M. Veloso, "Efficient physics-based planning: Sampling search via non-deterministic tactics and skills." in *AAMAS (1)*, 2009, pp. 27–33.

[36] 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.

[37] 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.

[38] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[39] 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.

[40] N. Widulle and O. Niggemann, "Using Reverse Reinforcement Learning for Assembly Tasks," in *PRL Workshop Series –Bridging the Gap Between AI Planning and Reinforcement Learning*, 2023.

[41] 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.

[42] 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.

[43] 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.

[44] 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. [Online]. Available: https://www.mdpi.com/2218-6581/11/6/137

[45] ——, "Alternative inverse kinematic solution of the ur5 robotic arm," in *Advances in Automation and Robotics Research*, H. A. Moreno, I. G. Carrera, R. A. Ramírez-Mendoza, J. Baca, and I. A. Banfield, Eds. Cham: Springer International Publishing, 2022, pp. 200–207.

[46] 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. [Online]. Available: https://doi.org/10.1115/1.4052805

[47] 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.

[48] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X20303979

[49] 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.

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

[51] D. M. Bodily, T. F. Allen, and M. D. Killpack, "Motion planning for mobile robots using inverse kinematics branching," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5043–5050.

[52] 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.

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

[54] 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.

[55] M. F. Barnsley, "Chapter II - metric spaces; equivalent spaces; classification of subsets; and the space of fractals," in *Fractals Everywhere (Second Edition)*, second edition ed., M. F. Barnsley, Ed. Academic Press, 1993, pp. 5–41. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780120790616500073

[56] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[57] K. P. Hawkins, "Analytic inverse kinematics for the universal robots ur-5/ur-10 arms," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:123980651