Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly

YUNSHENG TIAN, MIT CSAIL, USA
JIE XU, MIT CSAIL, USA
YICHEN LI, MIT CSAIL, USA
JIELIANG LUO, Autodesk Research, USA
SHINJIRO SUEDA, Texas A&M University, USA
HUI LI, Autodesk Research, USA
KARL D.D. WILLIS, Autodesk Research, USA
WOJCIECH MATUSIK, MIT CSAIL, USA

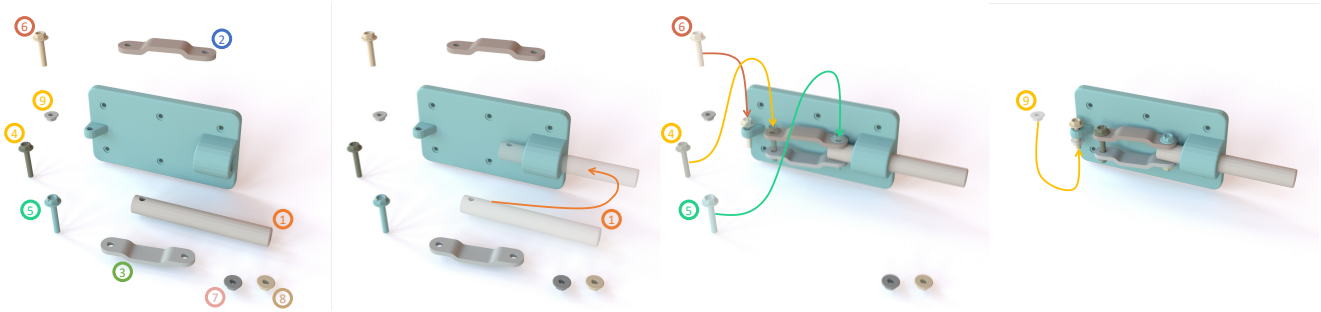

Fig. 1. We present a physics-based planning algorithm for assembly tasks. Our algorithm follows an assembly-by-disassembly routine and utilizes a custom physics-based simulation to efficiently explore the assembly path space. Our algorithm efficiently determines the feasible assembly order (colored numbers) for a multi-part assembly and searches for a physically realistic assembly motion path (colored curves) for each assembly step.

Assembly planning is the core of automating product assembly, maintenance, and recycling for modern industrial manufacturing. Despite its importance and long history of research, planning for mechanical assemblies when given the final assembled state remains a challenging problem. This is due to the complexity of dealing with arbitrary 3D shapes and the highly constrained motion required for real-world assemblies. In this work, we propose a novel method to efficiently plan physically plausible assembly motion and sequences for real-world assemblies. Our method leverages the assembly-by-disassembly principle and physics-based simulation to efficiently explore a reduced search space. To evaluate the generality of our method, we define a large-scale dataset consisting of thousands of physically valid industrial assemblies with a variety of assembly motions required. Our experiments on this new benchmark demonstrate we achieve a state-of-the-art success rate and the highest computational efficiency compared to other baseline algorithms. Our method also generalizes to rotational assemblies (e.g., screws and puzzles) and solves 80-part assemblies within several minutes.

Authors' addresses: Yunsheng Tian, MIT CSAIL, USA, yunsheng@csail.mit.edu; Jie Xu, MIT CSAIL, USA, jiex@csail.mit.edu; Yichen Li, MIT CSAIL, USA, yichenl@csail.mit.edu; Jieliang Luo, Autodesk Research, USA, rodger.luo@autodesk.com; Shinjiro Sueda, Texas A&M University, USA, sueda@tamu.edu; Hui Li, Autodesk Research, USA, hui.xylo.li@autodesk.com; Karl D.D. Willis, Autodesk Research, USA, karl.willis@autodesk.com; Wojciech Matusik, MIT CSAIL, USA, wojciech@csail.mit.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2022 Copyright held by the owner/author(s).
0730-0301/2022/12-ART278
https://doi.org/10.1145/3550454.3555525

CCS Concepts: • **Computing methodologies → Motion path planning**; *Physical simulation.*

Additional Key Words and Phrases: Assembly planning, physics-based planning, disassembly, dataset

ACM Reference Format:
Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D.D. Willis, and Wojciech Matusik. 2022. Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly. *ACM Trans. Graph.* 41, 6, Article 278 (December 2022), 15 pages. https://doi.org/10.1145/3550454.3555525

1 INTRODUCTION

Real-world objects, such as vehicles, furniture, and electronic devices, are often complex assemblies made up of hundreds or thousands of parts. Each part is designed using computer aided design (CAD) software and then arranged into a digital assembly piece by piece. Considerations for physical assembly, known as Design for Assembly (DFA), are critical to ensure fast, efficient, and high-yield manufacturing. Likewise, considerations for physical disassembly, known as Design for Disassembly (DFD), are key to ensuring individual parts can be recycled or reused at end-of-life. The ability to automatically plan the steps required to assemble and disassemble designs created in CAD is an enabling technology for a number of downstream applications. In the design phase, DFA checks can be performed to correct potential issues before manufacture [Melckenbeeck et al. 2020]; assembly planning for high mix, low volume

products can be greatly simplified with automatic generation of assembly instructions [Agrawala et al. 2003]; and at end-of-life, robotic disassembly systems [Ong et al. 2021] can be quickly adapted to recycle product components.

Despite significant research on assembly planning [Ghandi and Masehian 2015; Santochi et al. 2002], it remains a challenging problem for several reasons. Firstly, parts to be assembled can be of any shape — even common parts, such as screws, contain complex surface geometry. Secondly, real-world assemblies often contain tightly packed parts, requiring assembly planning approaches that work well in a constrained space. Finally, the lack of large-scale assembly datasets has limited evaluation to small hand-picked collections. How well existing methods generalize across large-scale datasets remains unknown.

To address this challenge, we introduce a novel physics-based method that takes an assembly-by-disassembly approach for sequence and motion planning with rigid assemblies. Given a CAD model in an assembled state, we use a custom physics-based simulation to disassemble individual parts and recover viable assemble plans (Figure 1). A key insight of our work is that physics-based simulation allows us to find disassembly motions using a discrete set of actions rather than exploring the full continuous 6D motion space. These discrete actions, represented as forces in canonical directions, can lead to successful motion when applied in physical simulation. For example, Figure 2 shows how a washer will slide along an inclined shaft even if the force is not applied directly along the disassembly direction. Using this approach our custom physics-based simulation overcomes the difficulty of navigating parts through narrow passages with geometric motion planning, allowing us to explore highly constrained motion spaces without sampling invalid states in most situations. To evaluate our approach and enable future research, we define a large-scale benchmark task for assembly planning consisting of thousands of physically valid assemblies — two orders of magnitude larger than previous benchmarks. We make the following contributions in this work:

- We introduce an accurate and efficient physics-based simulator that is customized for assembly.
- We propose a novel physics-based assembly-by-disassembly planning method for translational and rotational assembly motion for arbitrary-shaped assemblies.
- We define a large-scale dataset and benchmark for assembly planning including thousands of physically valid assemblies.
- We evaluate our method on the full dataset and show a state-of-the-art success rate, computational efficiency, and generalization on various types of assemblies scaling to hundreds of parts.

To facilitate future assembly planning research, our code and dataset are available at https://github.com/yunshengtian/Assemble-Them-All.

2 RELATED WORK

Assembly Sequence Planning (ASP). ASP arranges an optimal sequence in assembling all the components of a product and is a critical step for assembly system design [Rashid et al. 2012]. The ASP problem can be represented as a graph such as AND/OR graph [De Mello and Sanderson 1990] to establish feasible assembly sequences. Different optimization techniques are applied to find the optimal assembly sequence among the feasible sequences [Chen et al. 2006, 2008; Qin and Xu 2007; Sinanoğlu and Börklü 2005]. The premise of these methods is that precedence constraints between parts need to be known such that feasible sequences could be discovered easily. Otherwise, the precedence relationships need to be annotated by domain experts or derived from the CAD models [Niu et al. 2003; Su 2009]. Although many attempts have been made to reduce the search space, the number of potential assembly sequences can grow exponentially [Ramos et al. 1998] which makes identifying a feasible sequence impossible even for complex assemblies.

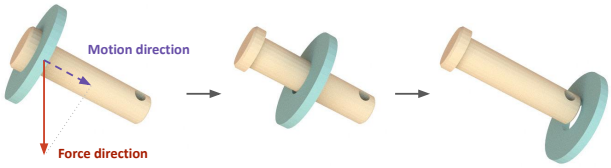

Fig. 2. Our physics-based planning method is motivated by the observation that approximate force directions can be used for disassembly. In this example, the washer can be disassembled from the shaft as long as the force applied has a positive projection along the true disassembly direction, assuming no friction exists.

The idea of assembly-by-disassembly has been proposed as an important strategy for speeding up ASP, because assembled parts have better defined precedence and motion constraints than disassembled parts [Homem de Mello and Sanderson 1991], reducing the search space. When all parts are rigid, a bijection exists between the assembly and disassembly sequences, meaning an assembly sequence can be obtained from the reverse order of its disassembly sequence with much less complexity. [Ghandi and Masehian 2015] The assembly-by-disassembly approach has been adopted in several applications in manufacturing and construction, including mechanical product assembly [Homem de Mello and Sanderson 1991; Wang et al. 2014], kit assembly [Zakka et al. 2020], and robotic additive construction [Huang et al. 2021]. However, if the assembling direction are unknown or the assembling motions are non-linear, then more sophisticated assembly path planning methods are needed to make sequence planning feasible, even following an assembly-by-disassembly strategy.

Assembly Path Planning (APP). APP computes penetration-free paths for adding parts to a subassembly, given the initial and target poses of the parts. Early works adopted exact geometric-reasoning methods that analyze part geometry to determine assembly directions [Wilson and Latombe 1994]. Halperin et al. [2000] presented a general framework for finding assembly motion called the motion space approach. Since geometric-reasoning approaches need to explicitly construct the Configuration space (C-space), they are computationally expensive for 3D space and non-linear motions.

With the development of general-purpose path planning algorithm in robotics, sampling-based approaches have been proposed to solve problems with a large number of parts [Masehian and Ghandi 2021]. Probabilistic Roadmap Method (PRM) [Kavraki et al. 1996] and Rapidly-exploring Random Tree (RRT) [LaValle et al. 1998] are

the two notable sampling-based algorithms. PRM, RRT, and their variants have been presented on assembly problems [Le et al. 2009; Sundaram et al. 2001; Zhang et al. 2020]. But they usually suffer from the 'narrow passage' problem [Hsu et al. 1999], especially in high-dimensional space where rotation needs to be considered.

Physics-based motion planning [Zickler and Veloso 2009] was developed as a competitive alternative to geometric-based motion planning and showed success with robotic navigation [Sucan and Kavraki 2011] and manipulation [Moll et al. 2017]. However, physics-based motion planning has not been fully explored for assembly where the motion is much more constrained. In our work, we demonstrate the benefit of using physics-based planning for efficient exploration of assembly paths to overcome the 'narrow passage' problem.

Physics-Based Simulation for Robotic Assembly. With the ability to create a virtual and low-cost replica of the real world and provide a fast robotic evaluation platform, physics-based simulation has been widely used in control policy learning [Andrychowicz et al. 2020; Brockman et al. 2016; Chen et al. 2021b] and motion planning [Sucan and Kavraki 2011; Zhou et al. 2014; Zickler and Veloso 2009].

Despite its prevalence in robotics, simulating robotic assembly tasks is still far from perfect. Simulating contact-rich assembly tasks, such as a screw and nut in Figure 4, requires accurately representing the object surface while efficiently resolving contact constraints. Most existing simulators [Coumans and Bai 2016; Makoviychuk et al. 2021; Todorov et al. 2012] rely on a simplified collision proxy of the object shape for efficient collision detection and solve the linear complementary problem for contact handling. However, such collision proxies sacrifice the resolution of complex geometry shapes and is only suitable for simple assembly tasks, such as peg-in-hole [Chebotar et al. 2019; Hou et al. 2020] or box stacking [Aleotti and Caselli 2009]. Another category of contact-rich simulators uses penalty-based contact models [Geilinger et al. 2020; Xu et al. 2021], but most only support collision detection between contact points and primitive shapes. Our simulation follows the line of penalty-based contact model, and is specialized for complex-shape assembly tasks via an extension of a signed-distance-field (SDF) collision detection. Concurrent to our work, Narang et al. [2022] announced a GPU-based simulator for assembly tasks, which also utilizes the SDF shape representation.

Robotic Assembly. Motion planning has been widely studied in robotic assembly for manufacturing [Chen et al. 2021a; Wan et al. 2017] and building construction [Hartmann et al. 2021; Huang et al. 2021], albeit in less realistic scenarios where planners are limited to structured environments. Reinforcement Learning (RL) [Sutton and Barto 2018], on the other hand, has been applied to more industrial and general-purpose assembly tasks [De Winter et al. 2021; Fan et al. 2019; Thomas et al. 2018; Yu et al. 2021] in unstructured environments [Luo and Li 2021]. However, the trained policies are usually task-specific and struggle with arbitrary unseen objects. Our real-world dataset and SDF-based simulation can aid RL-based methods in acquiring more generalized policies. Likewise, our physics-based planner can generate complex assembly motions that can directly guide robots or serve as demonstrations to learn from [Roldán et al. 2019; Zhu and Hu 2018].

3 PROBLEM FORMULATION

Given a collection of parts and their assembled state as input, our goal is to generate a sequence of physically realistic motion paths that position the parts into an assembled state, identical to the ground truth assembly.

Formally speaking, a state is defined as a vector s that encodes the position of the part in the state space S. In our problem, S can be either $\mathbb{R}^3$ for translational motion only or $SE(3)$ for both translational and rotational motion. For a given part i in an assembly consisting of M parts, a state s^i is defined as a valid state as long as it does not have penetration with other parts, assuming other parts are fixed. A state s^i is regarded as a disassembled state if the convex hull of the part geometry at s^i does not have collision with the convex hull that encloses the geometries of all other parts. A disassembly path P_D^i consists of a sequence of n valid states $\{s_0^i, ..., s_n^i\}$ that connects the assembled state s_0^i, given as input and a disassembled state s_n^i. Correspondingly, an assembly path $P_A^i = \{s_n^i, ..., s_0^i\}$ exists. Under these definitions, our problem aims at finding an ordered sequence of assembly paths for all M parts that assemble them from scratch to the ground truth assembly with all the states being valid.

We make the following key assumptions in this problem: 1) Assemblies are all composed of rigid parts, which makes assembly-by-disassembly feasible; 2) Gravity or other force constraints and manipulation constraints are not considered as we mainly address the assembly planning from the motion level. In other words, we assume the assembly process is fully-actuated; 3) We assume parts can be completely assembled or disassembled sequentially, though handling of subassemblies could be extended by grouping and treating subassemblies as single parts.

4 PHYSICS-BASED ASSEMBLY PLANNING

Searching for an assembly strategy to move multiple parts from a disassembled state to a target assembled state is challenging, especially when the assembly process requires highly accurate insertion or screwing operations. This is because in a typical assembly task, the final assembled state space is much more constrained than the initial disassembled state space. As a result, an extremely high computational cost is incurred when searching from the less constrained space to a constrained final state. Motivated by this, we solve the assembly problem in a reverse process, starting from the assembled state and searching for a disassembly solution, to reduce the overall search cost. We also leverage our custom physics-based simulator to further improve search efficiency.

Our main algorithm pipeline is illustrated in Figure 3. We first introduce our physics simulation for assembly in Section 4.1 as the key component to our algorithm. Next, following the assembly-by-disassembly strategy, we illustrate our disassembly path planning algorithm for two-part disassembly in Section 4.2, then our disassembly sequence planning algorithm for multi-part disassembly in Section 4.3. Finally, we present in Section 4.4 how to obtain the assembly sequence and path by reversing solutions from disassembly planning which completes our assembly-by-disassembly pipeline.

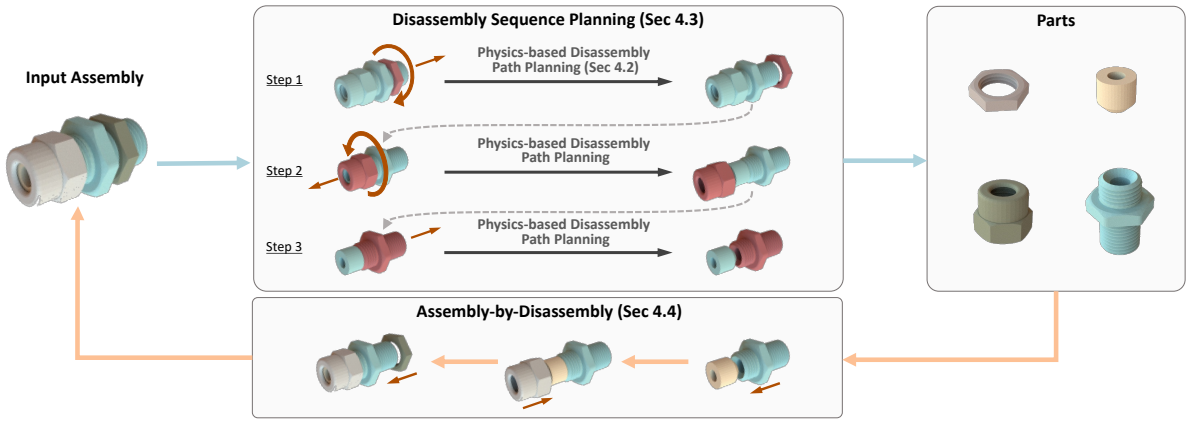

Fig. 3. Overview of our physics-based assembly planning algorithm. Given the target assembled state as input (left), our algorithm finds the assembly strategy in two stages: disassembly and assembly. In the disassembly stage (upper center), our disassembly sequence planning algorithm searches for a feasible disassembly order for different parts, using a physics-based disassembly path planning algorithm that plans collision-free paths to disassemble each part. In the assembly stage (lower center), the algorithm reverses the disassembly path to obtain an assemble solution from individual parts (right).

4.1 Physics Simulation for Assembly

The existing robotics simulators [Coumans and Bai 2016; Makoviychuk et al. 2021; Todorov et al. 2012] primarily rely on the convex hull decomposition of the object shape for efficient collision detection, which is not suitable for simulating complex geometry shapes with high concavity. To reliably simulate contact-rich assembly motion for downstream assembly planning, we build our physics-based simulation upon the rigid body simulator developed by Xu et al. [2021] and extend it with the SDF representation of collision shapes to support complex collision geometries.

Specifically, the simulator developed by Xu et al. [2021] is based on the reduced coordinate rigid body dynamics formulation of RedMax [Wang et al. 2019a], and its dynamics equations are implicitly integrated in time with BDF1 scheme (backward Euler). To handle collisions, first the contacting parts are detected between each pair of collision shapes, where the mesh vertices of the first collision shape are sampled as contact points, and a distance function associated with the second collision shape is used to compute the following relevant collision information for each sampled contact point:

$d,\ \dot{d}$	penetration distance and speed
$\vec{n}$	unit-length contact normal direction

Then for each detected contact pair, the contact forces are computed by a penalty-based contact model as below:

$$\vec{f_c} = (-k_n + k_d\dot{d})d\vec{n}, \tag{1}$$

where $\vec{f_c}$ is the contact normal force on each contact pair, k_n is the contact stiffness, and k_d is the contact damping coefficient. In our work, we assume no friction exists as friction does not affect the assembly motion under the assumption that every part is rigid. To efficiently acquire the required distance function, Xu et al.[2021] restrict one of each pair of collision shapes to be a primitive shape with an analytical distance function.

4.1.1 Signed Distance Field Collision Detection. To support contact handling between complex assembly geometries (e.g., screw with fine threads), we equip each collision shape with an SDF to provide the distance function. Being an implicit surface representation for arbitrary shape geometry, an SDF is a function $g(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ that maps a point in space to its closest distance to the represented shape surface (Figure 4). The distance is negative when the point is inside the geometry while the distance is positive if the point is outside.

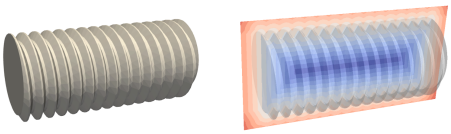

Fig. 4. We construct a signed distance field (right) for each collision shape (left). The right figure shows the level set of the cross-section signed distance field of a fine-thread screw.

Given the SDF of an object and the dynamics state $(\mathbf{x}, \dot{\mathbf{x}})$ of the contact point, the required collision information can be efficiently computed. The penetration distance $d = \min(g(\mathbf{x}), 0)$. The contact normal direction of the penetration is computed by the gradient of the SDF, i.e., $\vec{n} = \nabla g(\mathbf{x})$. The time derivative of the penetration distance is computed as $\dot{d} = \nabla g(\mathbf{x}) \cdot \dot{\mathbf{x}}$. And the relative tangential speed can be computed by projecting out the normal directional component from the relative speed between the two collision bodies $\vec{v}_t = \vec{v}_r(x) - (\vec{n} \cdot \vec{v}_r(x))\vec{n}$, where $\vec{v}_r(x)$ is the relative speed of two collision bodies at the contact point $\mathbf{x}$ which can be readily computed by rigid body simulator. For details on constructing the SDF grid, please refer to Appendix A.

4.2 Disassembly Path Planning

For a given part in the assembly state s_0, we look for a sequence of states $\{s_0, ..., s_n\}$ that removes the part from the rest of the assembly in a penetration-free manner. The state s_n here is a valid disassembled state. Different from other path planning tasks, such as navigation or stacking objects, which usually operate in free space with infinite valid solution paths, disassembly requires highly constrained motion, similar to passing through a narrow corridor. This makes

sampling-based geometric motion planning approaches less effective because of the low probability of obtaining a penetration-free state from random sampling. By contrast, humans rely on physics feedback to disassemble parts. As shown in Figure 2, even though we may apply forces that are not perfectly aligned with the ground-truth disassembly motion, we are able to infer the correct motion direction after the induced movement of the part is observed.

From this key observation, we propose a physics-based planner that efficiently plans the disassembly motion by leveraging feedback from physics. We formulate the disassembly path planning as a tree search problem, which starts from the assembled state and searches for a sequence of actions until a disassembled state has been found or some time/depth limitation of the search has been reached. The outline of our algorithm is illustrated in Algorithm 1.

Specifically, we adopt breadth-first search (BFS) and maintain a queue Q to store the states to be searched from. The search starts with the queue only containing the assembled state s_0. In each iteration, we dequeue a state s from Q, and loop over all actions a_i in a pre-defined action space A to generate child states of the disassembly tree. In our method, the action space consists of unit forces or torques of the given state space. More specifically, in translational-motion-only cases, $A = \{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$ which has 6 actions corresponding to 6 translational degrees of freedom (DoF); similarly, when rotational motion is enabled, A has 12 actions for 6 translational DoF and 6 rotational DoF. For each action $a_i \in A$, we continuously apply this action starting from the dequeued state s in physics-based simulation with a certain time step Δt, until the new state s_i becomes either a disassembled state or a similar state to one that has been searched in the past. If s_i is a disassembled state, then the disassembly path planning succeeds and we can obtain the path from the initial assembled state s_0 to s_i as a disassembly path which can be retrieved from backtracking the search tree from s_i. Otherwise, if s_i is detected to be a similar state to any previous state, we stop applying this action, enqueue the current state s_i, and continue searching with the rest of actions. Here we measure the similarity between two states by their translational and rotational distances. The translational distance is defined by the Euclidean distance $\|s_{a_t} - s_{b_t}\|$ between two states, where s_{a_t} and s_{b_t} are the linear components of the two states. The rotational distance is defined by the geodesic distance $\|\ln(s_{a_r}^{-1} s_{b_r})\|$ between the quaternions of two states, where s_{a_r} and s_{b_r} are the quaternion representations of the angular components of two states. Two states are similar if their translational and rotational distances are within thresholds δ_t and δ_r respectively.

As a result, BFS continuously explores the state space in a hierarchical manner and we enforce it to always explore novel states by checking state similarity. Note that the dynamics of our physics-based simulation constrains BFS to search in a physically valid subspace rather than the full state space which avoids the narrow passage problem in previous geometric motion planning methods. BFS is empirically quite effective because most real-world assemblies are designed to be assembled/disassembled within only one or a few sequential actions, which means that the depth of BFS is usually small.

ALGORITHM 1: Disassembly path planning guided by BFS

```
Input: Assembled state s_0, timeout t_max, max BFS depth d_max.
Output: A disassembly path P_D = {s_0, ..., s_n}.
Q = EmptyQueue();
Q.Enqueue(s_0);
while time t < t_max and BFS depth d ≤ d_max do
    s = Q.Dequeue();
    for a_i in action space A do
        s_i = s;
        do
            s_i = Simulate(s_i, a_i, Δt);
            if IsDisassembled(s_i) then
                return GetPath(s_0, s_i);
        while s_i is not similar to any other past states;
        Q.Enqueue(s_i);
return failed;
```

Our BFS-guided disassembly path planning method works well for general industrial assemblies as empirically proved by our evaluation. But for rare cases like disassembling parts from a long zig-zag path, BFS might be less efficient. In this case, we suggest a more sophisticated search algorithm such as Monte-Carlo Tree Search (MCTS) [Coulom 2006] along with carefully designed heuristics might be applied to speed up the search, though it might not perform equally well on simpler assemblies.

4.3 Disassembly Sequence Planning

Because of the internal precedence relationship between parts, most real world assemblies consist of multiple parts and require specific sequences to assemble/diassemble them. However, the precedence relationship is usually unknown beforehand. Without a carefully designed algorithm, the time complexity to figure out the correct disassembly sequence may grow up quadratically as the number of parts in an assembly increases. When coupled with the path planning algorithm, the overall running time for multi-part disassembly algorithm could be extremely slow [Ebinger et al. 2018].

We propose a method to efficiently plan the disassembly sequence by progressively expanded BFS, see Algorithm 2. Given the assembled states $\mathbf{s} = \{s_0^1, ..., s_0^M\}$ of all M parts, we look for an ordered sequence of disassembly paths $\mathbf{P}_D = \{\{s_0^{I_1}, ..., s_n^{I_1}\}, ..., \{s_0^{I_M}, ..., s_n^{I_M}\}\}$ that connect the assembled state and an disassembled state for each part and satisfy precedence relationships. Here, $I_1, ..., I_M$ correspond to ordered indices of the part sequence ranging from 1 to M. In each iteration, our sequence planner tries to disassemble each of the remaining parts of the assembly by our path planner from Algorithm 1. If any part is successfully disassembled, we append its disassembly path to the sequence $\mathbf{P}_D$. We repeat this procedure until all parts are disassembled or we reach the timeout.

Executing this procedure naively without limiting the BFS depth could be time-consuming due to the excessive time wasted on trying to disassemble parts that are blocked by other parts. Therefore, we take a progressive approach that starts with a shallow BFS and gradually increases the depth of BFS as the sequence planning progresses. For example, in the first iteration, we limit our path planning algorithm for disassembling all parts to have a maximal BFS depth

ALGORITHM 2: Disassembly sequence planning with Prog. BFS

Input: Assembled states $\mathbf{s} = \{s_0^1, ..., s_0^M\}$ of all M parts, path planning timeout $t_{\max}$, sequence planning timeout $T_{\max}$.
Output: An ordered sequence of disassembly paths $\mathbf{P}_D = \{P_D^{I_1}, ..., P_D^{I_M}\} = \{\{s_0^{I_1}, ..., s_n^{I_1}\}, ..., \{s_0^{I_M}, ..., s_n^{I_M}\}\}$.

$\mathbf{P}_D = \{\}$, max BFS depth $d_{\max} = 1$;
while time $t < T_{\max}$ **do**
  for part i in all remaining parts of the assembly **do**
    P_D^i = DisassemblyPathPlanning(s_0^i, $t_{\max}$, $d_{\max}$);
    if $P_D^i \neq$ failed **then**
      $\mathbf{P}_D$.Append(P_D^i);
      Remove part i from the assembly;
  if all M parts are disassembled **then**
    return $\mathbf{P}_D$;
  $d_{\max} = d_{\max} + 1$;
return failed;

$d_{\max} = 1$. Then, for each part it tries to disassemble, if our path planning algorithm fails to find a disassembly path with a single step of action application, we will temporarily stop trying this part and move on to other parts. There could be two reasons for this failure: 1) this part is blocked by other parts, i.e., conflict with the precedence relationships; 2) this part is not blocked but requires multiple actions to disassemble. Our progressive approach prevents us from wasting unnecessary effort in the first case while allowing us to try more actions in the later stage to handle the second case. Again, since most real-world assemblies are designed to be easily assembled/disassembled within one or a few actions, the first case is much more common and by early termination of the failed attempts we empirically observed a significant speed-up compared to working with unlimited BFS depth in Section 6.3.

4.4 Assembly-by-Diassembly

After the disassembly sequence and paths have been found, we can reverse them and connect with the initial states to construct the entire assembly sequence and paths through assembly-by-disassembly strategy, as illustrated in Algorithm 3. To simplify the notation we ignore the superscript I_i for part index. Each disassembly path P_D obtained from Algorithm 1 and 2 connects the assembled state s_{goal} and a disassembled state s_{dis}, but the path between s_{dis} and initial state s_{init} (which is specified by the user) is still missing. Different from the disassembly path, this path does not require highly constrained motion since both states are disassembled, we can easily apply a standard geometric motion planning algorithm such as RRT-Connect [Kuffner and LaValle 2000] to efficiently plan a collision-free path P_C between s_{init} and s_{dis}. Therefore, we can obtain the assembly path P_A from initial state s_{init} to assembled state s_{goal} by concatenating P_C and the reversed disassembly path P_D. By looping over the disassembly sequence reversely, we finally obtain an ordered assembly sequence containing all the assembly paths.

5 ASSEMBLY DATASET

To accurately evaluate our method we define a large-scale dataset for assembly planning. A focus of popular large-scale 3D shape datasets has been on synthetic assemblies that are semantically segmented [Chang et al. 2015; Mo et al. 2019] with part mobility labels [Hu et al. 2017; Wang et al. 2019b; Xiang et al. 2020; Yan et al. 2019]. In comparison, real world CAD assemblies contain detailed design, such as fastener stacks, and are segmented using different criteria, such as manufacturing process and ease of repair. We build upon the recent release of several realistic CAD assembly datasets [Jones et al. 2021; Koch et al. 2019; Willis et al. 2022, 2021] to define our dataset and benchmark for assembly planning. To the best of our knowledge, there does not exist a large-scale open-source dataset for assembly planning research.

We derive our dataset from assemblies created in Autodesk Fusion 360 and provided by Willis et al. [2022] as well as assemblies from Ebinger et al. [2018] and Zhang et al. [2020]. To adapt the data for use with assembly planning we perform several geometric pre-processing steps to produce watertight, unique, and normalized meshes suitable for accurate collision checking. We filter out assemblies where parts are in a non-assembled state, or in a state of overlap. Table 1 lists the total number of assemblies and their distribution by number of parts in each assembly. Figure 5 shows a random sampling of assemblies in the dataset. For a more detailed description of the geometric pre-processing steps on the dataset, please refer to Appendix B.1.

ALGORITHM 3: Assembly by disassembly

Input: Initial states $\mathbf{s}_{\text{init}} = \{s_{\text{init}}^1, ..., s_{\text{init}}^M\}$ and assembled states $\mathbf{s}_{\text{goal}} = \{s_{\text{goal}}^1, ..., s_{\text{goal}}^M\}$ of all M parts.
Output: An ordered sequence of assembly paths $\mathbf{P}_A = \{P_A^{I_M}, ..., P_A^{I_1}\} = \{\{s_{\text{init}}^{I_M}, ..., s_{\text{goal}}^{I_M}\}, ..., \{s_{\text{init}}^{I_1}, ..., s_{\text{goal}}^{I_1}\}\}$.

Compute ordered disassembly paths $\mathbf{P}_D$ from Alg. 2 given $\mathbf{s}_{\text{goal}}$;
$\mathbf{P}_A = \{\}$;
for i in $M, ..., 1$ **do**
  Obtain the disassembled state $s_{\text{dis}}^{I_i}$ from disassembly path $P_D^{I_i}$;
  Compute a collision-free path $P_C^{I_i}$ connecting $s_{\text{init}}^{I_i}$ and $s_{\text{dis}}^{I_i}$ by a standard motion planning algorithm (e.g., RRT-Connect);
  Connect assembly path $P_A^{I_i} = P_C^{I_i} + \text{Reverse}(P_D^{I_i})$;
  $\mathbf{P}_A$.Append($P_A^{I_i}$);
return $\mathbf{P}_A$;

Table 1. Dataset statistics.

# Parts	2	3-9	10-49	50-235	**Total**
# Assemblies	8776	2620	1468	106	**12970**

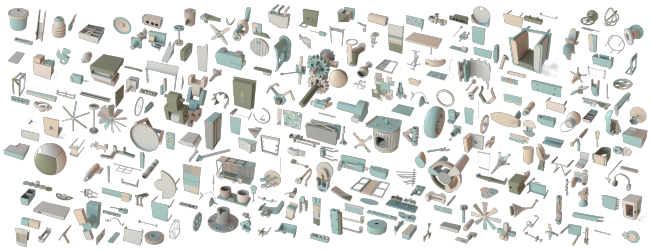

Fig. 5. Overview of our dataset containing thousands of physically valid assemblies suitable for use with assembly planning tasks.

6 EVALUATION

In this section, we perform experiments to evaluate our method on the proposed dataset with thousands of assemblies to demonstrate the advantage of our physics-based planning approach in terms of success rate and computational efficiency. For fairness, all baseline methods also follow the assembly-by-disassembly approach. Finally in Section 6.4, we show a naive extension of our method to solve more complicated assemblies that involve simultaneous movements of parts to disassemble.

We run all our experiments on Amazon EC2 instances (c5.24xlarge) with 96 vCPUs and 192G memory. The detailed hyper-parameter settings of all the methods can be found in Appendix C.2.

6.1 Baseline Methods

We choose the following path planning methods which are the most relevant baselines to compare: **RRT**: Rapidly-Exploring Random Tree [LaValle et al. 1998] is one of the most standard geometric path planning approaches, which requires an explicitly specified goal state. In our expriments, we use a randomly selected disassembled state as the goal for RRT. **T-RRT**: Targetless-RRT [Aguinaga et al. 2008] is a modified version of RRT that is specifically designed for disassembly planning, which does not require a specified specific goal state. **MV+T-RRT**: Mating Vector + Targetless-RRT, a hybrid approach proposed in [Ebinger et al. 2018] that first uses face normals of objects as "mating vectors" to try a straight-line disassembly, then if it fails, T-RRT is applied. This is the state-of-the-art approach in disassembly path planning to the best of our knowledge. **BK-RRT**: Behavioral Kinodynamic Rapidly-Exploring Random Trees [Zickler and Veloso 2009] is a classic physics-based planning algorithm similar to our method that uses a physics-based simulation to randomly explore new states. BK-RRT is shown to be successful in navigation and manipulation tasks but has not been evaluated in assembly or disassembly. More details of the baseline methods can be found in Appendix C.1.

6.2 Two-Part Assembly

We evaluate our path planning algorithm and baseline approaches on the two-part dataset and report the performance comparison in Table 2 (left). The reported success rates are calculated based on a 300s timeout, averaged across results from 6 random seeds. To further validate that our method generalizes to non-axis-aligned disassembly directions, we randomly rotate all the assemblies in the two-part dataset and report performance in Table 2 (bottom).

On this dataset, our approach achieves almost 100% success rate which is robust to various shapes and disassembly motions. Geometric baseline methods (RRT, T-RRT and MV+T-RRT) show decent success rates but would fail when disassembly motion is too constrained such as the task shown in Figure 6. In this example, the collision-free disassembly path should be strictly aligned with the axis of the cylinder, which is unlikely to be sampled from the whole motion space without assistance of a physics simulator. In contrast, both our approach and the physics-based baseline (BK-RRT) utilizes the physics simulator to mitigate the narrow passage problem. However, BK-RRT still fails in the task shown in Figure 6, that is because the random states explored by BK-RRT make it hard to keep moving in a single direction for a long period, thus result in a oscillatory motion of the ring on the cylinder. In our approach, the usage of BFS and state similarity check helps move part along one direction much easier. Another typical failure of the baseline methods is the difficulty of dealing with assemblies that require non-trivial rotational motions such as screws, as shown in Figure 7. For visualization on more results, please refer to Appendix D.

Table 2. Success rate (%) comparison on the entire two-part assembly dataset along with specific comparison on different rotational assemblies. Our method reaches almost 100% success rate on the two-part dataset and outperforms baseline methods by a large margin on rotational assemblies. The bottom row shows the results of our method on the same dataset with random rotations applied to the data.

Method	**Two-Part**	**Rotational**			
	Overall	Screw	Puzzle	Others	Overall
RRT	84.5	0.0	20.8	0.0	6.9
T-RRT	97.4	2.1	18.8	0.0	6.9
MV+T-RRT	97.8	12.5	18.8	0.0	10.4
BK-RRT	93.7	12.5	62.5	**81.3**	52.1
Ours	**99.8**	**75.0**	**87.5**	50.0	**70.8**
Ours (Rotated)	99.8	37.5	62.5	87.5	62.5

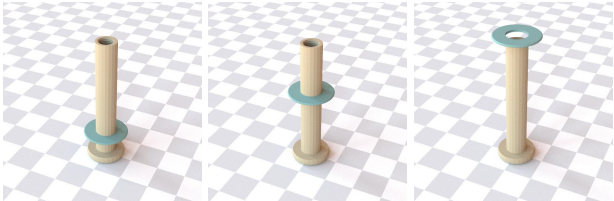

Fig. 6. A tightly-fitted two-part disassembly (left: assembled, right: disassembled). This can be easily solved by our method while being challenging for baseline methods due to the long and narrow passage.

6.2.1 Rotational Assembly. In this section, we specifically evaluate all algorithms on rotational assemblies, which are notoriously difficult for existing path planning approaches. In order to showcase different complexities of rotations, we prepare a rotational assembly dataset that consists of three categories of rotation to evaluate algorithm performance: 1) **Screw** for screws and nuts which are the most common rotational components in industrial assemblies; 2) **Puzzle** refers to puzzle assemblies that are mostly for entertainment purposes but are difficult to disassemble even for humans, gathered from [Zhang et al. 2020]; 3) **Others** for all other types of rotations, e.g., removing a ring along a curved tube. The rotational dataset has 24 assemblies in total (8 in each category) due to the scarcity of rotational assemblies. A complete overview of the rotational dataset can be found in Appendix B.2.

We report the results in Table 2 (right). The reported success rates are calculated based on a 600s timeout, averaged across results from 6 random seeds. From the results, we observe significantly better success rates of physics-based methods (i.e. BK-RRT and ours) in rotational disassembly path planning in all three categories of rotations. Our method outperforms other baselines by a large

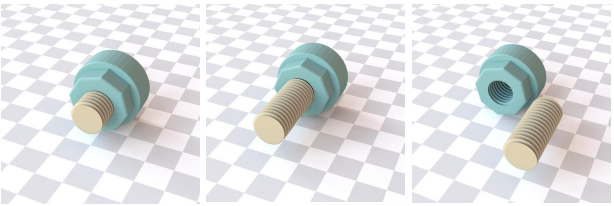

Fig. 7. Screw disassembly (left: assembled, right: disassembled). Our method successfully generates a long sequence of screwing motions and a short translation at the end to disassemble the screw from the nut.

margin in **Screw** and **Puzzle** assemblies while being worse than BK-RRT in the **Others** category. Figure 7 shows a screw example which can be successfully solved by our approach but fails on all other baseline methods.

For puzzle assemblies, Zhang et al. [2020] proposed a sampling-based geometric planning approach that combines with a learning-based geometric feature extractor to guide exploration. Their method takes hours of computation to find a disassembly path while we achieve a comparable success rate within several minutes on a subset of their puzzles. However, their approach is more specialized for solving more complicated puzzles and it is not designed for general types of assemblies.

6.3 Multi-Part Assembly

We evaluate our sequence planning algorithm on the portion of dataset with multi-part assemblies. We partition the multi-part dataset based on the assembly size into **Small**, **Medium** and **Large** categories corresponding to assemblies of 3-9, 10-49 and 50+ parts respectively. The statistics of each category are shown in Table 1. For baseline methods, we apply a similar disassembly sequence planning procedure as Algorithm 2 but without progressive BFS. For our method, we compare two variants: **Prog. BFS** as sequence planning with progressive BFS illustrated in Algorithm 2 and **Full BFS** as Algorithm 2 without limiting the max BFS depth.

We first compare the success rates of different algorithms in Table 3. The results are produced with a 120s timeout for each path planning attempt and a 7200s timeout for the entire sequence planning, averaged from 3 random seeds. Table 3 shows that our method (either Full BFS or Prog. BFS) outperforms other baselines on all different assembly sizes.

Furthermore, our method is capable of disassmbling complex many-part assemblies much faster than existing state-of-the-art approaches. For example, we successfully disassemble the entire 53-part Engine example (Figure 8) from [Ebinger et al. 2018] in 5 minutes while their MV+T-RRT takes hours to complete. We also empirically demonstrate in Table 4 that our sequence planning method guided by progressive BFS effectively cuts down the computation time by 3.5x on average compared to using the full BFS.

6.4 Sub-Assembly and Interlocking Assembly

Although most data in our dataset can be sequentially assembled, in this section, we extend our method to handle assemblies requiring simultaneous movements of multiple parts to disassemble. The first scenario is the use of sub-assemblies, where multiple parts form a group that must be disassembled together and follow the same disassembly motion. The second scenario is interlocking assemblies where multiple parts must follow different motions to free the assembly from the interlocked state. Here we describe a naive extension of our physics-based planning method to handle sub-assembly and interlocking assembly:

Given an M-part assembly, assuming at least m parts ($1 < m < M$) need to be moved to disassemble it, we need to figure out which m parts to move (sequence planning) and how these parts should move (path planning). For sequence planning, different from Algorithm 2 where it searches for a single part to disassemble, we extend it to search for all m-part combinations. For path planning, sub-assemblies can be treated as single parts and follow the same procedure in Algorithm 1, but for interlocking assemblies, instead of searching for single actions for single parts, we need to search for m actions to be applied to the m parts respectively.

Through such an exhaustive search combined with physics-based planning, Figure 9 shows an example where our extended method identifies the correct parts to disassemble along with the correct actions to apply. This is achieved by simultaneously applying a rotational torque around the z-axis (vertical axis) to the beige block and a positive translational force along the z-axis to the grey block.

Table 3. Success rate comparison (%) on multi-part assembly dataset. Our method consistently outperform baseline methods on all sizes of assemblies.

Method	**Multi-Part**			
	Small	Medium	Large	Overall
RRT	90.0	78.1	44.3	84.7
T-RRT	94.0	84.8	52.5	89.7
MV+T-RRT	94.2	85.2	53.8	90.0
BK-RRT	96.4	91.6	51.9	93.6
Ours (Full BFS)	**99.1**	96.8	71.1	97.6
Ours (Prog. BFS)	99.0	**97.3**	**76.7**	**97.9**

Table 4. Mean disassembly time per part (s) comparison on multi-part assembly dataset with different sizes of assemblies. Our proposed Prog. BFS gives large computational efficiency improvement compared to Full BFS.

	Small	Medium	Large	Overall
Full BFS	18.5	38.4	30.6	25.6
Prog. BFS	4.5	10.1	19.4	6.7

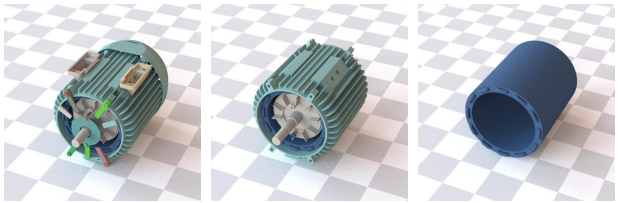

Fig. 8. 53-part engine disassembly (left: 53 parts left, middle: 11 parts left, right: 4 parts left). Our method with progressive BFS disassembles the engine completely within 5 minutes, an order of magnitude faster than current state-of-the-art approaches.

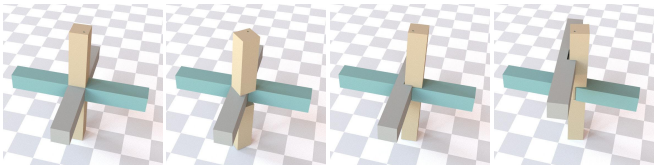

Fig. 9. A 3-part interlocking assembly disassembled by the movement of 2 parts (left: assembled, right: disassembled). Our extended physics-based planner discovers the correct disassembly strategy by simultaneously applying a counterclockwise rotational torque on the beige block and an upward vertical force on the grey block.

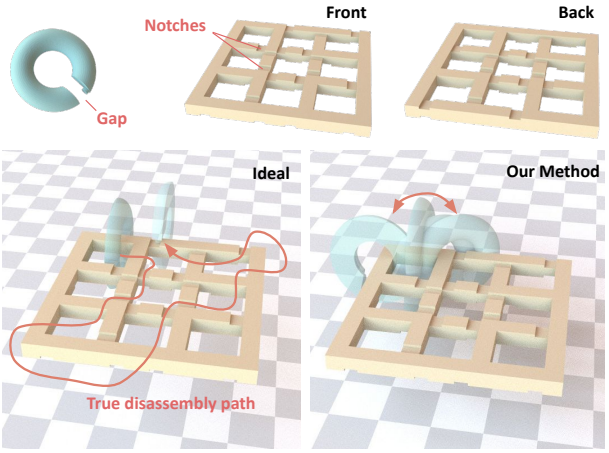

Fig. 10. A failure case of our method disassembling a maze (geometries shown in the top row). The goal is to disassemble the ring by navigating through multiple narrow notches of the maze grid (bottom left). Our method finds difficulty in aligning the gap of the ring with notches of the grid, so the ring gets stuck in the initial local region (bottom right).

Initially, the translational force applied on the grey block will be counteracted by the contact force from the beige block, but after the beige block rotates for 180°, it immediately pulls the grey block upwards to disassemble. In this case, the physical interactions between blocks are crucial to constrain and guide the disassembly motion towards the correct direction.

7 LIMITATIONS

In this section, we analyze the main failure cases of our method from the experimental results shown in Section 6.

Complex action sequences in path planning. Assemblies that require very complex sequences of actions could be time-consuming for our BFS-guided method to explore. For example, in Figure 10, we test our method on a challenging maze-like assembly from Zhang et al. [2020]. Here, the motion space is less constrained but the solution space is extremely narrow. It requires complex action sequences to navigate the ring through multiple tunnels by correctly aligning the gap of the ring with the notches of the maze grid.

Computational complexity on large assemblies. Both our method and baseline methods fail on some large assemblies. This is due to larger assemblies having much richer contacts that typically result in a super-linear increase in the time cost of both the simulation and sequence planning. Figure 11 shows several examples where all methods timeout after 2 hours.

Fig. 11. Large assemblies with several hundreds of parts and rich contacts where all tested methods reach the timeout of 2 hours.

The sequence planning complexity is $O(M^2)$ for M-part sequential assemblies. However, for using sub-assemblies or interlocking assemblies, the complexity will grow to $O(M!)$ due to the combinatorial search for multiple parts to disassemble together in each step. Therefore, our naive extension is by no means an efficient method for large assemblies that involve sub-assemblies or interlock. In this case, heuristics or learning-based methods can potentially be adopted to speed up the sequence search.

8 CONCLUSION AND FUTURE WORK

Planning for assemblies with arbitrary size, shape and motion is a long-standing and challenging problem. In this paper we have introduced a novel physics-based method for assembly planning. By providing a new method, custom simulation approach, and large-scale dataset we hope to enable future work that solves real world assembly and disassembly problems. We envision several key extensions would make assembly planning more general and efficient.

First, our assembly-by-disassembly strategy limits the assemblies to being rigid only. However, geometric-based approaches cannot handle deformable objects as well since they cannot model the physical deformation. Therefore, we believe it is interesting to explore further the direction of physics-based planning for generalization to deformable assemblies, such as the snap-fit assembly.

Second, it is essential to leverage geometric information beyond physical feedback in assembly planning. In contrast to the exhaustive search adopted by our method and baseline methods, humans can instantly infer plausible disassembly sequences and motions through vision and avoid wasting effort trying blocked parts or moving parts towards dead ends.

Finally, to facilitate research in robotic assembly, one exciting future work is adding robotic arms in our simulation to manipulate assemblies following the planned path generated by our approach. We believe it is promising to extend the capability of performing complex assembly autonomously and flexibly on real robots.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments in revising the paper. We thank Yijiang Huang, Tao Chen, and Tao Du for their valuable feedback on research idea formulation and Timothy Ebinger for providing part of the assembly models in Ebinger et al. [2018]. This work was supported in part by the National Science Foundation (CAREER-1846368).

REFERENCES

Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. 2003. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 828–837.

Iker Aguinaga, Diego Borro, and Luis Matey. 2008. Parallel RRT-based path planning for selective disassembly planning. *The International Journal of Advanced Manufacturing Technology* 36, 11 (2008), 1221–1233.

Jacopo Aleotti and Stefano Caselli. 2009. Efficient planning of disassembly sequences in physics-based animation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 87–92.

OpenAI : Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012* (2015).

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. 2019. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 8973–8979.

Guanlong Chen, Jiangqi Zhou, Wayne Cai, Xinmin Lai, Zhongqin Lin, and Roland Menassa. 2006. A framework for an automotive body assembly process design system. *Computer-Aided Design* 38, 5 (2006), 531–539.

Hao Chen, Weiwei Wan, and Kensuke Harada. 2021a. Planning to Build Soma Blocks Using a Dual-arm Robot. In *2021 IEEE International Conference on Development and Learning (ICDL)*. IEEE, 1–7.

Tao Chen, Jie Xu, and Pulkit Agrawal. 2021b. A Simple Method for Complex In-hand Manipulation. *Conference on Robot Learning* (2021).

Wen-Chin Chen, Pei-Hao Tai, Wei-Jaw Deng, and Ling-Feng Hsieh. 2008. A three-stage integrated approach for assembly sequence planning using neural networks. *Expert Systems with Applications* 34, 3 (2008), 1777–1786.

Rémi Coulom. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*. Springer, 72–83.

Erwin Coumans and Yunfei Bai. 2016. Pybullet, a python module for physics simulation for games, robotics and machine learning. (2016).

LS Homem De Mello and Arthur C Sanderson. 1990. AND/OR graph representation of assembly plans. *IEEE Transactions on robotics and automation* 6, 2 (1990), 188–199.

Joris De Winter, Ilias EI Makrini, Greet Van de Perre, Ann Nowé, Tom Verstraten, and Bram Vanderborght. 2021. Autonomous assembly planning of demonstrated skills with reinforcement learning in simulation. *Autonomous Robots* 45, 8 (2021), 1097–1110.

Timothy Ebinger, Sascha Kaden, Shawna Thomas, Robert Andre, Nancy M Amato, and Ulrike Thomas. 2018. A general and flexible search framework for disassembly planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3548–3555.

Yongxiang Fan, Jieliang Luo, and Masayoshi Tomizuka. 2019. A learning framework for high precision industrial assembly. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 811–817.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact. *ACM Trans. Graph.* 39, 6, Article 190 (Nov. 2020), 15 pages.

Somaye Ghandi and Ellips Masehian. 2015. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-Aided Design* 67 (2015), 58–86.

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

Valentin Noah Hartmann, Andreas Orthey, Danny Driess, Ozgur S Oguz, and Marc Toussaint. 2021. Long-horizon multi-robot rearrangement planning for construction assembly. *arXiv preprint arXiv:2106.02489* (2021).

L.S. Homem de Mello and A.C. Sanderson. 1991. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation* 7, 2 (1991), 228–240. https://doi.org/10.1109/70.75905

Zhimin Hou, Jiajun Fei, Yuelin Deng, and Jing Xu. 2020. Data-efficient hierarchical reinforcement learning for robotic assembly control applications. *IEEE Transactions on Industrial Electronics* 68, 11 (2020), 11565–11575.

David Hsu, Jean-Claude Latombe, Rajeev Motwani, and Lydia E Kavraki. 1999. Capturing the connectivity of high-dimensional geometric spaces by parallelizable random sampling techniques. In *Advances in randomized parallel computing*. Springer, 159–182.

Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. 2017. Learning to predict part mobility from a single static snapshot. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Yijiang Huang, Caelan R Garrett, Ian Ting, Stefana Parascho, and Caitlin T Mueller. 2021. Robotic additive construction of bar structures: Unified sequence and motion planning. *Construction Robotics* 5, 2 (2021), 115–130.

Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G. Kim, and Adriana Schulz. 2021. AutoMate: A Dataset and Learning Approach for Automatic Mating of CAD Assemblies. *ACM Transactions on Graphics (TOG)* 40, 6 (2021).

Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 9601–9611.

James J Kuffner and Steven M LaValle. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 2. IEEE, 995–1001.

Steven M LaValle et al. 1998. Rapidly-exploring random trees: A new tool for path planning. (1998).

Duc Thanh Le, Juan Cortés, and Thierry Siméon. 2009. A path planning approach to (dis) assembly sequencing. In *2009 IEEE International Conference on Automation Science and Engineering*. IEEE, 286–291.

Jieliang Luo and Hui Li. 2021. A Learning Approach to Robot-Agnostic Force-Guided High Precision Assembly. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2151–2157.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. 2021. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ellips Masehian and Somayé Ghandi. 2021. Assembly sequence and path planning for monotone and nonmonotone assemblies with rigid and flexible parts. *Robotics and Computer-Integrated Manufacturing* 72 (2021), 102180.

Ine Melckenbeeck, Sofie Burggraeve, Bart Van Doninck, Jeroen Vancraen, and Albert Rosich. 2020. Optimal assembly sequence based on design for assembly (DFA) rules. *Procedia CIRP* 91 (2020), 646–652.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. 2019. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 909–918.

Mark Moll, Lydia Kavraki, Jan Rosell, et al. 2017. Randomized physics-based motion planning for grasping in cluttered and uncertain environments. *IEEE Robotics and Automation Letters* 3, 2 (2017), 712–719.

Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, et al. 2022. Factory: Fast Contact for Robotic Assembly. *arXiv preprint arXiv:2205.03532* (2022).

Xinwen Niu, Han Ding, and Youlun Xiong. 2003. A hierarchical approach to generating precedence graphs for assembly planning. *International Journal of Machine Tools and Manufacture* 43, 14 (2003), 1473–1486.

SK Ong, MML Chang, and AYC Nee. 2021. Product disassembly sequence planning: state-of-the-art, challenges, opportunities and future directions. *International Journal of Production Research* 59, 11 (2021), 3493–3508.

YF Qin and ZG Xu. 2007. Assembly process planning using a multi-objective optimization method. In *Proceedings of the 2007 IEEE international conference on mechatronics and automation, ICMA*, Vol. 4303610. 593–598.

Carlos Ramos, Joao Rocha, and Zita Vale. 1998. On the complexity of precedence graphs for assembly and task planning. *Computers in industry* 36, 1-2 (1998), 101–111.

Mohd Fadzil Faisae Rashid, Windo Hutabarat, and Ashutosh Tiwari. 2012. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology* 59, 1 (2012), 335–349.

Juan Jesús Roldán, Elena Crespo, Andrés Martín-Barrio, Elena Peña-Tapia, and Antonio Barrientos. 2019. A training system for Industry 4.0 operators in complex assemblies based on virtual reality and process mining. *Robotics and computer-integrated manufacturing* 59 (2019), 305–316.

Marco Santochi, Gino Dini, and Franco Failli. 2002. Computer aided disassembly planning: state of the art and perspectives. *CIRP Annals* 51, 2 (2002), 507–529.

Cem Sinanoğlu and H Rıza Börklü. 2005. An assembly sequence-planning system for mechanical parts using neural network. *Assembly Automation* (2005).

Qiang Su. 2009. A hierarchical approach on assembly sequence planning and optimal sequences analyzing. *Robotics and Computer-Integrated Manufacturing* 25, 1 (2009), 224–234.

Ioan A Sucan and Lydia E Kavraki. 2011. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics* 28, 1 (2011), 116–131.

Sujay Sundaram, Ian Remmler, and Nancy M Amato. 2001. Disassembly sequencing using a motion planning approach. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, Vol. 2. IEEE, 1475–1480.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. 2018. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3524–3531.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

Weiwei Wan, Kensuke Harada, and Kazuyuki Nagata. 2017. Assembly sequence planning for motion planning. *Assembly Automation* (2017).

Hui Wang, Yiming Rong, and Dong Xiang. 2014. Mechanical assembly planning using ant colony optimization. *Computer-Aided Design* 47 (2014), 59–71.

Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. 2019b. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8876–8884.

Ying Wang, Nicholas J Weidner, Margaret A Baxter, Yura Hwang, Danny M Kaufman, and Shinjiro Sueda. 2019a. REDMAX: Efficient & flexible approach for articulated dynamics. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–10.

Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2022. JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

Randall H Wilson and Jean-Claude Latombe. 1994. Geometric reasoning about mechanical assembly. *Artificial Intelligence* 71, 2 (1994), 371–396.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. 2020. SAPIEN: A SimulAted Part-based Interactive ENvironment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. 2021. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Proceedings of Robotics: Science and Systems*. Virtual. https://doi.org/10.15607/RSS.2021.XVII.008

Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver van Kaick, Hao Zhang, and Hui Huang. 2019. RPM-Net: Recurrent Prediction of Motion and Parts from Point Cloud. *Annual Conference on Computer Graphics and Interactive Techniques Asia (SIGGRAPH Asia)* 38, 6 (2019), 240:1–240:15.

Mingxin Yu, Lin Shao, Zhehuan Chen, Tianhao Wu, Qingnan Fan, Kaichun Mo, and Hao Dong. 2021. RoboAssembly: Learning Generalizable Furniture Assembly Policy in a Novel Multi-robot Contact-rich Simulation Environment. *arXiv preprint arXiv:2112.10143* (2021).

Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. 2020. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9404–9410.

Xinya Zhang, Robert Belfer, Paul G Kry, and Etienne Vouga. 2020. C-Space tunnel discovery for puzzle path planning. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 104–1.

Hongkai Zhao. 2005. A fast sweeping method for eikonal equations. *Mathematics of computation* 74, 250 (2005), 603–627.

Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. 2014. Boxelization: Folding 3D Objects into Boxes. *ACM Trans. Graph.* 33, 4, Article 71 (Jul 2014), 8 pages.

Zuyuan Zhu and Huosheng Hu. 2018. Robot learning from demonstration in robotic assembly: A survey. *Robotics* 7, 2 (2018), 17.

Stefan Zickler and Manuela M Veloso. 2009. Efficient physics-based planning: sampling search via non-deterministic tactics and skills.. In *AAMAS (1)*. Citeseer, 27–33.

A SDF CONSTRUCTION IN SIMULATION

We construct the SDF grid for an object using the Fast Sweeping Algorithm [Zhao 2005] once at the initialization stage of the simulation. The Fast Sweeping Algorithm computes the distance function at the grid points around the object surface, and then updates the remaining grids with eight sweeps. Given the SDF grid, the distance of an arbitrary point in the space can acquired in $O(1)$ by trilinearly interpolating among surrounding grid cells and the gradients are computed in $O(1)$ via finite-differencing. Our assembly dataset contains thousands of objects with dramatically different sizes and also contains objects having different sizes in different dimensions (e.g. long poles). In order to capture the surface texture details of those objects while keeping the algorithm as fast and low-memory as possible, we use an adaptive SDF grid with cell size $\min(0.05, L_i/20)$ in each dimension i for an object, where L_i is the length of the object along the i-th dimension and assuming each assembly is scaled to fit in a 10x10x10 unit cube.

B DATASET DETAILS

B.1 Dataset Pre-Processing

Our dataset pre-processing consists of several steps to adapt the previous CAD assembly datasets for assembly planning. Specifically, in each assembly that contains two or multiple parts (meshes), we sequentially do the following steps:

(1) Load meshes from the source datasets with correct transforms to the assembled states applied.
(2) Remove non-watertight meshes since the distance query is poorly defined on non-watertight meshes.
(3) Remove duplicate meshes with the same geometry and position as other meshes, which is a rare design mistake.
(4) Remove meshes that overlap more than 10% with other meshes because assembly planning will less likely give a plausible result for such a large penetration. The overlap percentage is computed by sampling points inside the mesh volume and checking the ratio of points contained by other meshes.
(5) Remove thin meshes, which are difficult for SDF-based collision detection to compute distance correctly. This step is optional if the simulation is not based on SDF for collision detection and can handle thin structures well. In our implementation, we first compute the mesh's oriented bounding box (OBB). Next, we filter out meshes whose thinnest edge of OBB is smaller than 1% of the largest *scale* of the whole assembly, where the *scale* is defined by the length of the largest edge of the assembly's bounding box.
(6) Remove all other parts except the largest *connected* subset of the assembly. For example, all single floating parts will be removed. This step makes the assembly data more meaningful for evaluating assembly planning methods because parts floating in the air are already disassembled or very close to being disassembled. In our implementation, two *connected* parts are defined as having a collision between their convex hulls.
(7) Normalize all the remaining meshes in an assembly to fit in a 10x10x10 bounding box. This makes all assemblies have similar scales, thus benchmarking assembly planning methods on the whole dataset becomes much easier, considering setting a single set of hyper-parameters of the algorithm that generalize to the entire dataset (e.g., collision detection threshold, state similarity threshold, step size of RRT).
(8) Remove assemblies that are not disassemblable as rigid parts. This is done manually since an oracle does not exist to check if an assembly is disassemblable. This step is not guaranteed to be highly accurate considering the manual nature and dataset size.
(9) Subdivide the meshes such that the largest edge of the mesh is no larger than 0.5. This is because our simulation relies on a point-based contact model and does not support edge-edge collision detection yet. Increasing the density of mesh vertices helps the simulation handle contacts more robustly. This step is optional if the simulation can handle edge-edge collision well.

Note that we do not remove meshes with slight overlaps (penetration) with other meshes. This is because assembly CAD models designed by human designers typically have small penetration errors between parts. Rather than requiring the assembly model to be perfectly penetration-free, we set an adaptive threshold for collision detection based on the amount of initial penetration (for both our method and baseline methods). This allows path planners to generate plausible disassembly paths while some slight penetration exists.

B.2 Rotational Dataset Overview

Figure 12, 13 and 14 show a complete overview of our rotational dataset, which consists of rotational assemblies that require various types of assembly motion.

C EXPERIMENTAL SETUP

C.1 Baseline Methods

In this section, we describe some of our baseline methods in more details:

RRT [LaValle et al. 1998]: RRT iteratively grows a tree in the state space to reach the user-defined goal state. In Each iteration it either samples a random state in the space or chooses the goal state and then extends towards it from the nearest node in the existing tree. However, it requires an explicitly specified goal state, which can be an arbitrary disassembled state in disassembly path planning. Therefore, we use a randomly selected disassembled state as the goal for RRT.

T-RRT [Aguinaga et al. 2008]: Instead of choosing a random disassembled goal state like RRT, when expanding the tree, Targetless-RRT takes as the goal the nearest state outside the bounding box of the rest of the assembly. This goal heuristic is empirically more suitable for disassembly than random.

BK-RRT [Zickler and Veloso 2009]: See Algorithm 4 for a complete illustration of BK-RRT in disassembly path planning.

C.2 Hyper-Parameters

In this section, we present all hyper-parameters used in experimental evaluation. Specifically, Table 5 summarizes the main hyper-parameters used in our physics-based simulation, Table 6 shows the

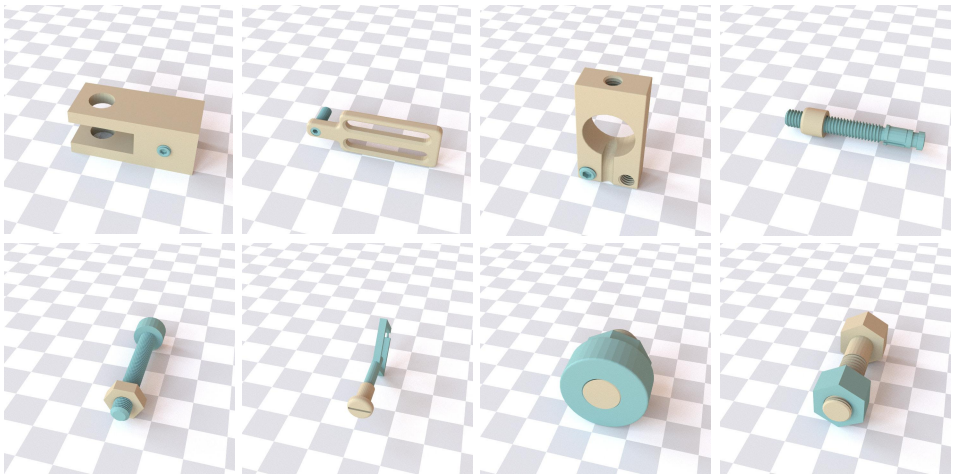

Fig. 12. **Screw** category of the rotational dataset.

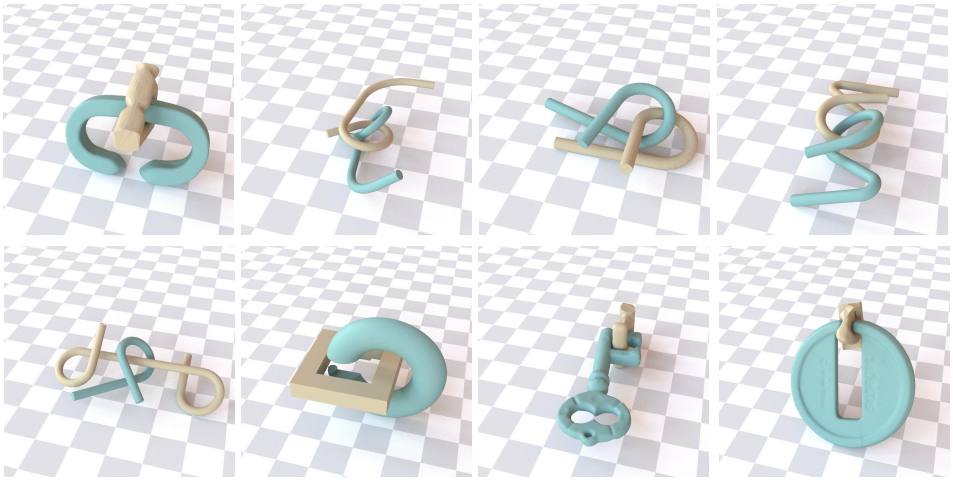

Fig. 13. **Puzzle** category of the rotational dataset.

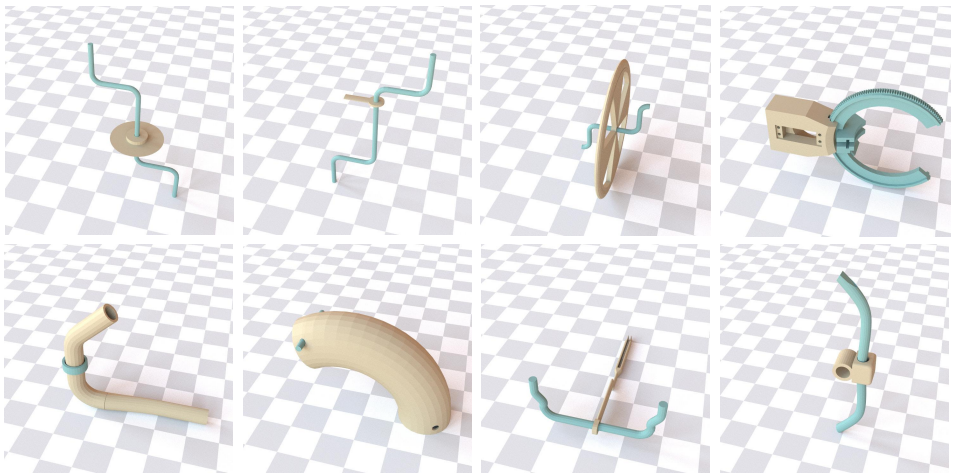

Fig. 14. **Others** category of the rotational dataset.

ALGORITHM 4: BK-RRT for Disassembly Path Planning

Input: Assembled state s_0, timeout $t_{\max}$, simulation time step Δt.
Output: A disassembly path $P_D = \{s_0, ..., s_n\}$.

```
T = EmptyTree();
T.AddNode(s_0);
while t < t_max do
    s_r = SampleRandomState();
    s_i = NearestNeighbor(T, s_r);
    a_i = RandomAction();
    s_{i+1} = Simulate(s_i, a_i, Δt);
    T.AddNode(s_{i+1});
    T.AddEdge(s_i, s_{i+1}, a_i);
    if IsDisassembled(s_{i+1}) then
        return T.GetPath(s_0, s_{i+1});
return failed;
```

Table 5. Hyper-parameters of physics-based simulation.

Name	Value
Contact stiffness k_n	1e6
Contact damping coefficient k_d	0
Simulation time step	1e-3

Table 6. Hyper-parameters of physics-based path planners.

Name	Value
Path planning time step Δ_t	1e-1
Penetration threshold for collision detection	0.01
Force/torque magnitude of each action	100
State similarity threshold (translation) δ_t	0.05
State similarity threshold (rotation) δ_r	0.5

Table 7. Hyper-parameters of geometric-based path planners.

Name	Value
Step size of tree extension	0.01
Maximum penetration allowed	0.01
Goal probability of T-RRT	0.2

hyper-parameters for physics-based path planners (our method and BK-RRT), and Table 7 shows the hyper-parameters for geometric-based path planners (RRT, T-RRT, MV+T-RRT).

D MORE RESULTS

See Figure 15 for more assembly motion produced by our method which have a low success rate on other baseline methods.

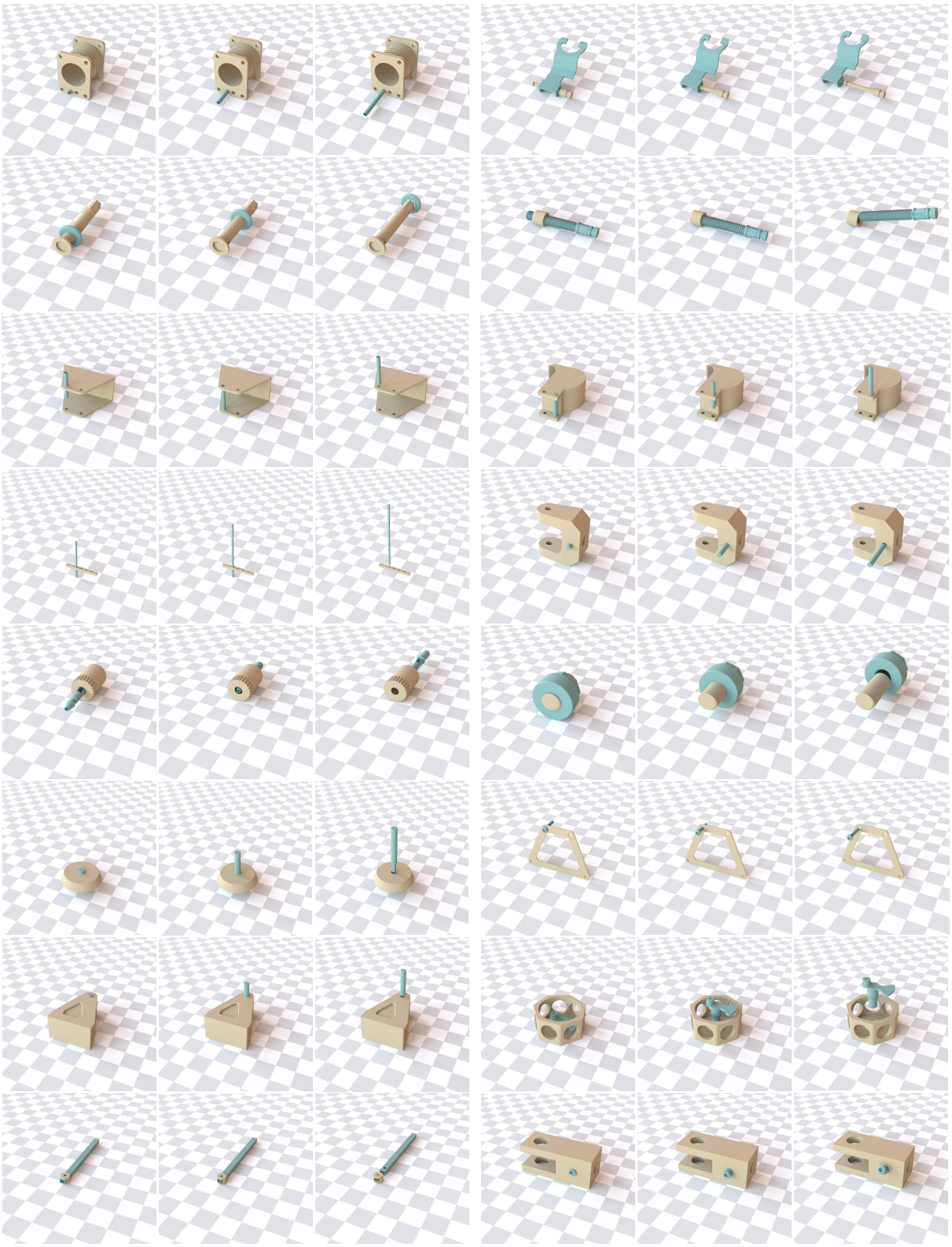

Fig. 15. More successful examples of two-part disassembly motion produced by our method while failing on baseline methods.